The Application of Harmony Search in Computer Vision

Jaco Fourie

Computer Science

University of Canterbury

A thesis submitted for the degree of

PhilosophiæDoctor (PhD)

February 2011

Abstract

The harmony search algorithm was developed in 2001 as a heuristic optimisation algorithm for use in diverse optimisation problems. After its introduction it was extensively used in multiple engineering disciplines with great success. In order to demonstrate the value of harmony search in computer vision applications I developed four novel algorithms based on harmony search that efficiently solves three problems that are commonly found in computer vision, namely visual tracking, visual correspondence matching and binary image restoration. Computer vision is a large discipline that includes solving many different kinds of optimisation problems. Many of these optimisation problems are discontinuous with derivative information difficult or impossible to come by. The most common solution is to use population based statistical optimisation algorithms like the particle filter, genetic algorithms, PSO, etc. but harmony search has never been investigated as a possible alternative. This is surprising since harmony search has been shown to be superior to these methods in several other engineering disciplines.

I therefore aim to show that harmony search deserves to be included in the computer vision researcher's toolbox of optimisation algorithms through the introduction of four novel algorithms based on harmony search that solve three diverse problems in computer vision. First the harmony filter (HF) is introduced as a visual tracking algorithm that is shown to be superior to the particle filter and the unscented Kalman filter (UKF) in both speed and accuracy for robust tracking in challenging situations. The directed correspondence search (DCS) algorithm is then introduced as a solution to the visual correspondence problem. Finally, two algorithms, counterpoint harmony search (CHS) and largest error first harmony search (LEFHS), are introduced for the blind deconvolution of binary images.

Comparative results from these algorithms are very promising. The harmony filter was compared with the particle filter and the UKF both of which have been extensively used in visual tracking. In challenging situations consisting of rapid and erratic target movement, extended periods of total and partial occlusion and changing light conditions, the HF proved to be more accurate and faster on average than both the particle filter and the UKF. Under various conditions I show that the HF is at least 2 times faster than a UKF implementation and 4 times faster than a particle filter implementation (using 300 particles).

While there are fewer algorithms specialising in the blind deconvolution of binary images, CHS and LEFHS were compared with a current state-of-the-art method and proved to be more robust to noise and more accurate. LEFHS is the only algorithm currently available that can recover a 24×12 binary image using blind deconvolution to 100% accuracy without putting constraints on the *point spread function* (blurring kernel).

During the development of these algorithms several valuable insights into the inner workings of harmony search were discovered. In each application harmony search had to be adapted in a different way and with each new adaptation a deeper understanding of the advantages of harmony search is revealed. Knowing which components may be modified without degrading performance is key to adapting harmony search for use in diverse problems and allows one to use harmony search in situations it was not originally designed for without losing its superior performance. These insights and the adaptation strategies that they lead to are the main contribution of this thesis.

Acknowledgements

This thesis and research that led to it would not have been possible without the constant support of my supervisors Dr Steven Mills and Dr Richard Green. Without their willingness to read through this thesis over their Christmas holidays it would not have finished when it did and I greatly appreciate the effort they put into this.

Equally so, I would like to express my deepest appreciation for my family. Without their encouragement I would certainly not have endured these past three years.

Finally and most importantly, none of this would have been possible without the power of the Lord my saviour. All glory be to God almighty!

The financial assistance of the National ICT Innovation Institute (NZi3) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not to be attributed to the NZi3.

Contents

List of Figures

List of Algorithms

List of Tables

Glossary and Nomenclature

Symbols

$\mathbf{f}$	Bold font indicates f is a multidimensional vector
$\mathbf{A}$	Capital bold font identifies $\mathbf{A}$ as a matrix
$\bar{\mathbf{f}}$	The mean of the vector $\mathbf{f}$
$\mathbf{x}_{t\|t}$	Shorthand for $p(\mathbf{x}_{0:t}\|\mathbf{z}_{0:t})$
$\nabla f(\mathbf{x})\|_{\mathbf{x}_0}$	The Jacobian matrix of f evaluated at $\mathbf{x}_0$
$corr(\mathbf{f_1}, \mathbf{f_2})$	The sample correlation coefficient between $\mathbf{f_1}$ and $\mathbf{f_2}$
$E(X)$	The expected value of X
$\alpha \propto k$	α is proportional to k
$A \Rightarrow B$	A implies B
$r \backsim U(-1, 1)$	r is distributed according to $U(-1, 1)$
$\|\mathbf{f_1} - \mathbf{f_2}\|$	the Euclidean distance between $\mathbf{f_1}$ and $\mathbf{f_2}$
$\|\Delta\|$	the cardinality of the set Δ
$\|\|x\|\|$	absolute value of x
LB_i	The lower bound of the i'th component in the search space
UB_i	The upper bound of the i'th component in the search space
$\mathcal{N}(m, \sigma)$	Normal distributed random number with mean m and standard deviation σ
$U(-1, 1)$	Uniform distributed random number between -1 and 1

Glossary

CHS	Counterpoint Harmony Search

DCS	Directed Correspondence Search
DE	Differential Evolution
DHS	Differential Harmony Search
DLHS	Dynamic Local Best Harmony Search
ECR	Ensemble Consideration Rate
EHS	Explorative Harmony Search
EKF	Extended Kalman Filter
FW	Fret Width
GHDE	Global Harmony Differential Evolution
GHS	Global Best Harmony Search
HF	Harmony Filter
HM	Harmony Memory
HMCR	Harmony Memory Consideration Rate
HMM	Hidden Markov Model
HMS	Harmony Memory Size
HS	Harmony Search
IS	Island Size
LEFHS	Largest Error First Harmony Search
NM-SA	Nelder-Mead Simplex Algorithm
PAR	Pitch Adjustment Rate
PDF	Probability Density Function
PSF	Point Spread Function
PSHS	Particle Swarm Harmony Search
PSO	Particle Swarm Optimisation
SA	Simulated Annealing
SAHS	Self-adaptive Harmony Search
SHS	Simplex-Harmony Search

SQP	Sequential Quadratic Programming
SSD	Sum of Square Differences
TER	Total Energy Regularization
TVR	Total Variation Regularization
UKF	Unscented Kalman Filter
UPF	Unscented Particle Filter

1

Introduction

> Our age of anxiety is, in great part, the result of trying to do today's jobs with yesterday's tools ...
>
> – *Marshal McLuhan*

The subject and main aim of this thesis is to introduce the harmony search algorithm (HS) as a useful and efficient algorithm for use in computer vision applications. Harmony search falls into the category of population based evolutionary algorithms and was designed as a general optimisation algorithm. It has been used with great success in many engineering disciplines including vehicle routing, the design of water networks and structural engineering [1–3]. Due to this success it has become a well known alternative to other popular evolutionary algorithms like genetic algorithms (GA), ant colony optimisation (ACO) and particle swarm optimisation (PSO)[1] [1, 4, 5].

The field of computer vision is filled with problems that can be modelled as classical optimisation problems, for example, the visual tracking problem and the restoration of images degraded by noise and blurring. It is not therefore surprising that evolutionary algorithms like GAs, SA and PSO have been extensively used in solving these problems [6–11]. What is surprising is that even though HS has been shown to be superior to many GA's, SA and PSO in several studies, it was not used in computer vision until the start of this study [1, 2, 12–17].

Harmony search is a relatively new *metaheuristic* optimisation algorithm first introduced by Dr. Z.W. Geem in 2001 [4]. The word *metaheuristic* is formed from the

[1] PSO is commonly considered a paradigm in swarm intelligence but is included in the general collection of evolutionary algorithms.

word heuristic, meaning to *find* or *search* through trial-and-error, and the prefix *meta*, which in this context means *above* or *higher*. So unlike heuristics that use trail-and-error methods to solve a specific problem, metaheuristics use higher level techniques to solve a general class of problems effectively.

As is often the case with evolutionary metaheuristics, HS was inspired by a natural phenomenon. In this case the methods used by professional musicians to collaboratively improvise new harmonies was taken as the inspiration for the harmony search algorithm. An analogy between improvisation and optimisation was constructed and an algorithm was designed to mimic the way a musician uses short-term memory and past experiences to lead her to the note that results in the most pleasing harmony when played together with the other musicians. Chapter 2 gives a detailed explanation of HS theory and development as well as implementation details and recommendations on parameter settings.

In order to demonstrate the value of HS in computer vision applications I developed four novel algorithms based on HS that efficiently solve three problems that are commonly found in computer vision, namely *visual tracking*, *visual correspondence matching* and *binary image restoration*. I show that by adapting HS to these specific problems one can create algorithms that perform as well as, or in some cases significantly better than, the current state-of-the-art.

1.1 Motivation

The field of computer vision aims to provide computers systems with the ability that human beings have to sense their surroundings by using their visual system. Our visual system allows us to recognise thousands of different objects almost instantly by combining the shape, texture and colour information that our visual system give us. Our visual system is amazingly robust in gathering this information and can still provide the necessary information we need for recognition even during dynamically changing light conditions, rapid movement of the object and severe deformation of the object. Through our visual system we are able to synchronise the movements of our body allowing us to perform complex tasks that require pinpoint visual accuracy like threading a needle or catching a ball.

Current computer systems cannot hope to match the performance of the human visual system but recent advances in computer vision have led to breakthroughs in industrial automation, robotic navigation and diagnostic medicine that would not have been possible otherwise. In industrial automation robots are now able to, by using computer vision techniques, correctly recognise manufactured parts, pick them up without causing damage and accurately guide them to the next stage of the manufacturing process [18]. By using computer vision and digital cameras, robots are now able to visually recognise landmarks and use them to build a map of its surroundings allowing them to navigate in much the same way that humans do [19, 20]. Modern computer vision has led to many advances in medical imaging allowing doctors to see inside the body like never before and make accurate diagnoses without having to resort to invasive surgeries [21, 22].

These are only some of the areas in which computer vision is being extensively used to further technology. Many technological disciplines have only recently started using computer vision and researchers are constantly finding new uses for these techniques. One reason for the slow adoption of computer vision is that it has been historically plagued by computationally expensive algorithms that are often only useful when done in real-time. To some extent this problem has been solved by modern processor technology enabling real-time performance from algorithms that were previously too computationally expensive. This explains the recent popularity of computer vision techniques but while processors became faster, so also did the quality of digital still and video cameras.

Digital cameras are becoming more and more ubiquitous and the demand for the processing of image data has grown almost as quickly as processing power has. The demand for quick and accurate computer vision algorithms has therefore not diminished with advances in processing speed. With the growing popularity of digital video surveillance everywhere in the world, the demand is likely larger now than its ever been before.

To meet this demand we need algorithms that can process an image and produce a result within the required accuracy in the least amount of time possible. Many of the optimisation algorithms in computer vision are designed to consider all possibilities and calculate the optimal solution every time. However, this approach can be very computationally expensive even to the point where it becomes practically useless. Often,

though, the application only demands a solution of sufficient accuracy and the exact solution is not necessarily required if such a result can be produced quickly enough.

This is the reason that metaheuristic algorithms are often used in computer vision systems that require sufficiently accurate solutions in the least amount of time possible. Metaheuristic algorithms cannot guarantee that the exact original solution will be found but are very good at quickly processing many potential solutions in a bounded amount of time and return the best possible solution given limited computational resources.

As previously mentioned, HS is a modern metaheuristic algorithm that has shown itself to be superior to many other commonly used metaheuristics. I therefore propose that based on the success of other metaheuristics in computer vision that HS can be used with equal or greater success. Direct application of HS in the same way that GAs, SA and PSO have been applied is possible and would likely lead to marginally better results but as we will see in Section 2.6, HS has the advantage of being easily adapted and augmented for use in specific problems leading to significantly better performance. My aim is then to take advantage of HS's structure and build novel heuristic algorithms based on HS but designed for specific use in computer vision applications.

1.2 Research Objectives

The primary objective of this thesis is to show that harmony search and harmony search based algorithms can be successfully applied to many problems commonly found in computer vision applications. The aim is to introduce harmony search to the computer vision community through a series of algorithms based on harmony search optimisation. These algorithms are designed to solve some commonly encountered problems in computer vision like robust visual tracking, visual correspondence matching and image restoration.

A secondary goal is to illustrate, through the development of these algorithms, that harmony search can easily be adapted to solve a much wider range of problems and therefore is genuinely valuable in the field of computer vision.

1.3 Contributions

Throughout the course of this study my research has led to several peer-reviewed publications that outline the main contributions of this thesis. The main contributions are the development of four novel algorithms for use in computer vision as listed below.

Harmony Filter The harmony filter (HF) was developed for the robust visual tracking of an arbitrary target through a video sequence. It is based on the improved harmony search algorithm (see Section 2.6.2) but was adapted for use in visual tracking. The HF is able to robustly track an object in real-time under very challenging conditions. The speed and accuracy of the HF was compared in challenging conditions with that of the popular Kalman Filter, particle filter and unscented particle filter. I show that HS is superior in both speed and accuracy and can robustly track in conditions that cause other algorithms to fail. The HF has led to three publications, namely in *Image and Vision Computing* ([23]), in the proceedings of the *23rd International Image and Vision Computing New Zealand* conference ([24]) and as a chapter in the book *Recent Advances in Harmony Search Algorithm* ([25]). The HF is the topic of Section 3.4.

Directed Correspondence Search The directed correspondence search (DCS) algorithm is another harmony search based algorithm and was designed to find the best visual correspondence set between the visual features of two images. The initial results showed rapid convergence to a reasonable correspondence set indicating that this algorithm can be effectively used to quickly create initial solutions for other established algorithms like RANSAC. The DCS algorithm adds the *swap operator* and *uniqueness constraint* to harmony search. This allows harmony search to solve any labelling problem and is not necessarily specific to the correspondence problem. The DCS algorithm was published in the proceedings of the *24rd International Image and Vision Computing New Zealand* conference (see [26]). More detail on this algorithm is given in Section 4.3.

Counterpoint Harmony Search Counterpoint Harmony Search (CHS) was developed for the restoration of binary images corrupted by noise and unknown blurring. This process, called blind deconvolution, is especially challenging when applied to binary images and perfect recovery is rarely achieved. CHS uses a

novel operator called the *cadenza* operator to adapt harmony search to the unique search space of this problem and allows it to perfectly recover the original image. In order to avoid the many local optima found in this particular problem CHS uses a novel distributed data structure to maintain diversity of solutions. The CHS algorithm was published in the proceedings of the *International Conference on Audio, Language and Image Processing* (see [27]). Details on the CHS algorithm can found in Section 5.3.

Largest Error First Harmony Search A major limitation of CHS is that it is only capable of restoring images corrupted by binary point spread functions (PSFs). This makes it of very limited use since non-binary PSFs are the norm in real images. In order to address this limitation I developed the largest error first harmony search (LEFHS) algorithm. LEFHS is able to reliably recover binary images degraded by noise and arbitrary blurring (binary and non-binary PSFs) to 100% accuracy. The speed of the recovery was also improved and LEFHS is significantly faster than CHS even when recovering images degraded with more complicated PSFs. LEFHS was compared with CHS and another leading blind deconvolution algorithm and was found to be superior to both. More details on LEFHS can be found in the proceedings of the *25th International Image and Vision Computing New Zealand* conference (see [28]). LEFHS is also discussed in detail in Section 5.4.

1.4 Thesis outline

The rest of this thesis investigates and discusses the harmony search algorithm and shows how it can be used to solve three commonly found problems in computer vision. In the next chapter I introduce harmony search as a metaheuristic optimisation algorithm. The musical inspiration behind harmony search is explored and I show how this analogy is used to develop a practical optimisation algorithm. Some theory is then given followed by practical implementation details. The rest of the chapter introduces several attempts by other researchers to improve on the original algorithm and also introduces several hybrid algorithms that use harmony search as the basis.

In Chapter 3 the visual tracking problem is discussed and I show how it can be interpreted as an optimisation algorithm. The harmony filter is then presented as a

novel harmony search based solution to the tracking problem. The harmony filter is thoroughly explained and I conclude the chapter by comparing tracking results from the harmony filter with that of other popular tracking algorithms like the Kalman filter, particle filter and the unscented particle filter.

The same general pattern is followed in Chapter 4 but instead of visual tracking the focus is now on the visual correspondence problem. As before an overview is given of current leading approaches to this problem followed by a section describing how one can model the visual correspondence problem as an optimisation problem. The harmony search based DCS algorithm is then introduced as a solution followed by a detailed description of DCS. Implementation details are again given and I conclude with results and analysis.

The blind deconvolution of binary images is topic of Chapter 5. In this chapter two algorithms are introduced, namely CHS and LEFHS. Both are harmony search based solutions to the blind deconvolution problem. The chapter starts with an overview of the current leading approaches to blind deconvolution of specifically binary images and then focusses on how this can be modelled as an optimisation algorithm. CHS is then introduced as a harmony search based solution with the usual explanations and implementation details following. The limitations of CHS is also explored, most notably the restriction to binary PSFs. With this restriction in mind LEFHS is introduced as a harmony search based alternative to CHS. LEFHS does not suffer from the same limitations that CHS suffers from and also performs faster making it an alternative that is of much more practical use. The chapter ends with results from both CHS and LEFHS that is then compared with other leading approaches.

In the last chapter I conclude with some final observations. In light of the algorithms previously introduced I critically evaluate harmony search as a practical algorithm for use in computer vision applications. I argue that, given the success that harmony search has shown in several computer vision applications, harmony search should be considered more often when designing algorithms for use in computer vision systems.

Throughout this thesis the main theme is adapting harmony search so it can be effectively used in a variety of computer vision applications. While the four novel algorithms, developed during this research, are stated as a main contribution of this work, the methods developed to adapt harmony search are equally significant. During the

development of these algorithms many insights were gained relating to the inner workings of harmony search and the relationship that its components have with each other. As we will learn in the chapters that follow, understanding these relationships is key to adapting harmony search in such a way that it maintains its speed and accuracy. This understanding leads to other areas of investigation including the analysis of parameter sensitivity, exploratory power in different search spaces, efficient convergence detection, theoretical advantages and others, all of which are discussed in this thesis.

2

Harmony Search

> The task that a symbol system is faced with, then, when it is represented with a problem and a problem space, is to use its limited processing resources to generate possible solutions, one after another, until it finds one that satisfies the problem defining test. If the symbol system had some control over the order in which potential solutions were generated, then it would be desirable to arrange this order of generation so that actual solutions would have a high likelihood of appearing early. A symbol system would exhibit intelligence to the extent that it exceeded in doing this. Intelligence for a system with limited resources consists in making wise choices of what to do next. ...
>
> – *Newell and Simon, 1976, Turing Award Lecture*

This chapter gives a detailed explanation of the harmony search (HS) algorithm. This includes an investigation of the algorithm's ability to explore a given search space and also the stochastic derivative that can be calculated for any objective function using harmony search. Later in the chapter different strategies for optimising harmony search's performance is explored and the sensitivity of the algorithm to these optimisations is evaluated. It then concludes with an investigation of various recent adaptations of harmony search based on combining harmony search with other optimisation algorithms to create a hybrid algorithm.

2.1 Heuristic Optimisation

The Hungarian mathematician George Polya was famous for his work in heuristics and defines it as "The study of the methods and rules of discovery and invention" [29]. Unlike more traditional optimisation methods that attempt to calculate the optimal solution directly from the search space, heuristic methods traverse the search space looking for the optimal solution using these *rules of discovery and invention*. Different heuristics define different rules on how to choose which branches in the state space will likely yield acceptable solutions and should be explored first.

Generally, there are two reasons why one would use heuristic methods as opposed to the standard approach by means of differential calculus or brute force enumeration [30].

1. The problem may not have an exact solution due to inherent ambiguities in the problem statement or available data. This is often the case in computer vision because visual scenes are often ambiguous with many equally valid interpretations of the real location of objects in a scene. Incomplete data of a scene due to, for example, only viewing from one perspective, leads to further ambiguities.

2. Even when the problem may have an exact solution the computational burden of finding this solution may be too great to be practical. In many computer vision problems we see combinatorially explosive growth in the size of the state space as the size of the images increases. For example, the number of possible solutions to the image deconvolution problem (see Chapter 5) grows exponentially with the size of the degraded image.

Additionally, heuristic methods may be favoured over other methods when the objective function is not easily differentiated, when it is not convex or when it is discontinuous. Many optimisation algorithms require that the function must be continuous over the region of interest and that the derivative be available. When the objective function is not convex over the region of interest it becomes likely that algorithms based on derivative information (gradient based search) will fail to find the global optimum but instead get stuck in local optima even when the function is continuous and derivatives are available. Most heuristic methods do not require that the objective function be smooth and are designed to escape local optima when the objective function is non-convex.

Heuristic algorithms are often designed to mimic some behaviour found in nature. An example of this is the *survival of the fittest* principle used for selection and diversity through mutation, both of which inspired the genetic algorithm originally developed by Holland in 1975 [31]. Another example is the annealing process used in metallurgy that inspired the simulated annealing algorithm which is an adaptation of the Metropolis-Hastings algorithm [32]. Other examples include the self organising cooperative strategy that ants in a colony use to navigate (ant colony optimisation) [33] and the way insects or birds synchronise to move together in a swarm or flock (particle swarm optimisation or PSO) [34]. Similarly, the harmony search algorithm was inspired by the behaviour of improvising musicians playing together in a band.

In the taxonomy of heuristic algorithms harmony search uses a population based or multiple agent heuristic similar to genetic algorithms, ant colony optimisation and particle swarm optimisation. Population based methods maintain a population of candidate solutions and typically update a whole set of them simultaneously. The advantage of this is that a better representation of the search space is achieved and that the probability of getting stuck in local optima is reduced. However, unlike most population based algorithms harmony search does not update the whole population simultaneously. As we will see in the next section only one member of the population is updated during each iteration[1].

2.2 Musical Improvisation as a Heuristic

Harmony search was introduced by Geem et al. in 2001 as a heuristic optimisation algorithm for use in general optimisation problems [4, 5]. The algorithm is inspired by the process musicians use to find the most pleasing harmony when playing together in a band. For the sake of the following analogy we call this process improvisation.

Consider a band of musicians each playing a different instrument and each trying to find the note that, when played simultaneously with the rest of the band, results in perfect harmony. For sake of simplicity lets say that the band consists of only three musicians and that each musician can only play one of three different notes. This situation is illustrated in Figure 2.1. Let us further suppose that we can measure the

[1] In many texts on optimisation the term *generation* is used to indicate one iteration of the optimiser. I will mostly refer to it as an iteration but will use the term *generation* when referring to its use in other texts.

quality of the harmony produced using some function, f. Since there are three musicians this will be a function of three variables with each variable representing the note that each musician played. The analogy between improvisation and optimisation becomes clear when we realise that finding the perfect harmony is the same as optimising f which is called the *objective function* in optimisation literature. Since f measures the quality or *fitness score* of the resulting harmony, f is also sometimes referred to as the *fitness function*. Each possible input to the objective function represents an attempt by the musicians to find the perfect harmony and is represented as a vector of notes called a *solution vector*. All possible combinations of notes from each of the three musicians form a pool of possible solutions called the *search space* which is very small for this example and only contains $3^3 = 27$ possible solution vectors.

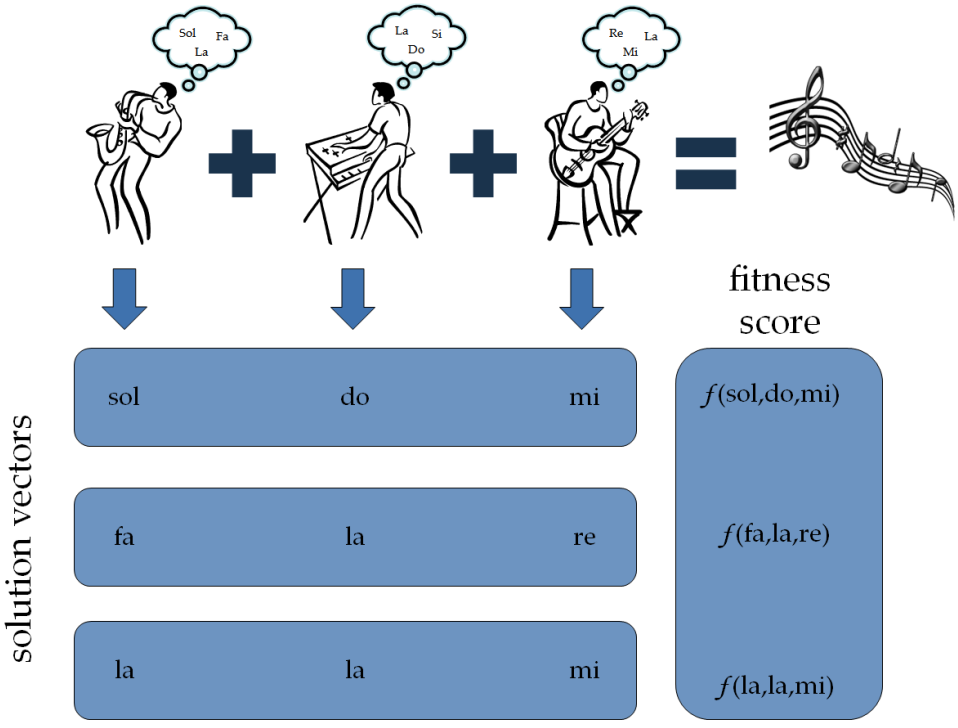

Figure 2.1: This diagram illustrates the analogy between finding musical harmony and optimisation. The thought cloud above each musician shows the vocabulary of notes available to him.

Thus far the analogy only shows that improvisation can be interpreted as an optimisation problem but this analogy also leads to an algorithm that allows one to effectively

solve the optimisation problem. During improvisation, each musician chooses and plays a note from his vocabulary and then listens to the resulting harmony to measure its quality. If the harmony is pleasing the musician will remember that note and use it to base future improvisations on. For example the next note he tries might be one half-tone lower or higher than the one that resulted in pleasing harmony when played with the rest of the band. However, if the resulting harmony is not pleasing the musician will not remember that note and will likely choose a completely different note in the next attempt. After several attempts each musician has built up a memory of good solutions that are combined and adapted to form ever better harmonies which eventually leads to the band finding the perfect combination of notes that result in perfect harmony.

The harmony search algorithm uses a data structure called the harmony memory (HM) to emulate the short term memory of good harmonies used by the musicians. The HM is a matrix of solutions vectors and their fitness scores as measured by the objective function. The number of solution vectors in the HM is controlled by the harmony memory size (HMS) parameter which is set before initialisation. The HMS controls the number of solutions that are simultaneously considered during optimisation and can have a large effect on the speed of the algorithm. In Section 2.5.3 we will discuss how the HMS should be chosen and its effect on the performance of the algorithm.

New solution vectors are improvised based on solutions in the HM using one of three methods namely *memory consideration*, *pitch adjustment* and *random selection*. These are similar to the options available to the musicians during improvisation. They can either choose a note from one of the ones previously memorised (memory consideration), pick a note randomly from the instrument's range of playable notes (random selection) or they could slightly adjust one of the memorised notes to become a neighbouring note (pitch adjustment). Musicians decide how to improvise based on experience and training but the harmony search algorithm improvises according to three parameters namely the *harmony memory consideration rate* (HMCR), the *pitch adjustment rate* (PAR) and the *fret width* (FW)[1]. Together with the HMS[2]these form the set of four parameters that determine the speed and accuracy of harmony search.

Once an improvisation has been made it is scored using the objective function. If the score is better than the lowest scoring member in the HM then the lowest scoring

[1]The fret width was formally known as the bandwidth (BW) and it is still sometimes referred to as such. Dr Geem has since updated the terminology to better reflect the musical analogy [5].

[2]For all abbreviations see the glossary on page xii.

member in HM is removed and replaced by the new improvisation. This process of improvisation and update of the HM is repeated until the maximum number of iterations are reached or until convergence is detected (convergence detection is investigated in Section 2.5.2).

The section that follows gives a detailed explanation of harmony search by way of a simple example. However from the steps already mentioned we already have an overview of the algorithm. The main steps of harmony search are as follows. These are the steps that we will work through in the section that follows.

1. Initialise the HM with possible solutions from the entire search space of the objective function.
2. Improvise a new solution vector using solutions from the HM and random search.
3. If the new improvisation is better than the worst solution in the HM, replace the worst solution with the new improvisation.
4. Check for convergence. If convergence or stopping criteria have not been met go to step 2, else go on to step 5.
5. Return the highest scoring member in the HM as the optimal solution.

2.2.1 A Simple Example

Consider the following polynomial

$$f(x, y, z) = ||(x - 11)^3|| + (y - 4)^2 + ||z - 7|| \ ,$$

where $|| \bullet ||$ indicates the absolute value. It is clear to see that the minimum of this function is found at $\{x, y, z\} = \{11, 4, 7\}$[1]. If we let harmony search find this minimum point and choose the HMS to be $N = 3$, a random initialisation of the HM is illustrated in figure 2.2. In this simple example the objective function is the polynomial itself and is directly evaluated to score solution vectors.

In the example of Figure 2.2 only 3 solution vectors are kept in the HM at the same time. Each solution is evaluated by the objective function in order to find the

[1]Solutions are not restricted to integer solutions but integer examples are used here to simplify calculation.

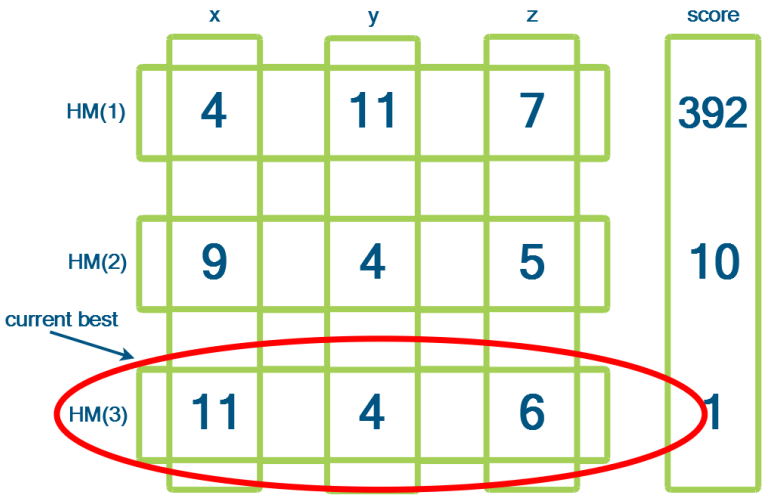

Figure 2.2: This HM is randomly initialised with three solution vectors. Each column represents one component of the objective function and each row is a solution vector. The score is indicated as a separate column and since we are minimising in this example the member with the lowest score is the best member in the HM.

best and worst solution in the HM. Since we are trying to minimise the function the best solution would have the lowest function value and the worst, the highest. After initialisation the best solution vector is thus the third one, namely $\{11, 4, 6\}$.

Once the HM has been initialised a new solution vector is improvised component-by-component using either memory consideration, pitch adjustment, or random selection. The relative frequency with which these three methods are chosen is controlled by the previously mentioned parameters namely the HMCR and the PAR. Both of these values are chosen before initialisation and lie between 0 and 1. The mechanics of the improvisation step are as follows.

First a uniform random number between 0 and 1 is generated and if it is greater than the HMCR random selection is chosen as the means of improvisation and the component value is chosen randomly from the set of possible values. If it falls below the HMCR the HM is taken into consideration and another random number between 0 and 1 is generated. If this new number falls above the PAR then memory consideration is chosen as the means of improvisation and the component value is chosen randomly from the set of values in the HM for that component (see pseudo-code in Algorithm 2.1 for details). If the random number falls below the PAR a random value from the HM for that component is chosen and adjusted using the pitch adjustment operator. The

pitch adjustment operator slightly changes the value of the component in an attempt to improve previous solutions or escape local optima by using the following equation

$$x'_{new} = x_{new} + r \times FW, \tag{2.1}$$

where $r \backsim U(-1, 1)$ and FW is the fret width parameter that controls the maximum variation in pitch adjustment and is set before initialisation[1]. x_{new} is the newly improvised solution vector and x'_{new} is the solution vector after pitch adjustment. The pitch adjustment operator is similar to the mutation operator found in genetic algorithms and serves the same purpose.

Define M as the number of components (also known as decision variables which there are 3 of in this simple example) in the search space and UB_i and LB_i as the upper and lower bound value of the $i'th$ component. Let $U(0, 1)$ be a random number between 0 and 1 generated from a uniform distribution and x_i be the $i'th$ component of the solution vector x. The pseudo-code for the improvisation process is given in Algorithm 2.1.

input : The initialised HM
output: The new improvisation $x_{new} = [x'_0, ..., x'_M]$

```
1  foreach i ∈ [1, M] do
2      if U(0,1) > HMCR then                          // random selection
3          x'_i ← LB_i + r × (UB_i − LB_i)             // where r ∽ U(0,1)
4      else                                           // memory consideration
5          x'_i ← x^j_i                                // where j ∽ U(1, HMS)
6          if U(0,1) ≤ PAR then                        // pitch adjustment
7              x'_i ← x'_i + r × FW                    // where r ∽ U(−1,1)
8          end
9      end
10     x_new[i] ← x'_i
11 end
```

Algorithm 2.1: The improvisation step

An example of all three improvisation methods being used to generate a new solution vector is illustrated in Figure 2.3. The first component, x, is generated using memory consideration. This means that the x-value of the new improvisation has to be ∈

[1]For a nomenclature of symbols used see page xii

$\{4, 9, 11\}$ and in this case 11 was chosen. The second component, y, is improvised using pitch adjustment. The value from the second solution vector (4) was selected and a small random offset is applied to create the improvised value of 5. The third component, z, is improvised using random selection and can be any value in the set of all possible values for z in the search space. As seen in the example, the value does not need to be present in the HM and in this case the value 12 was chosen.

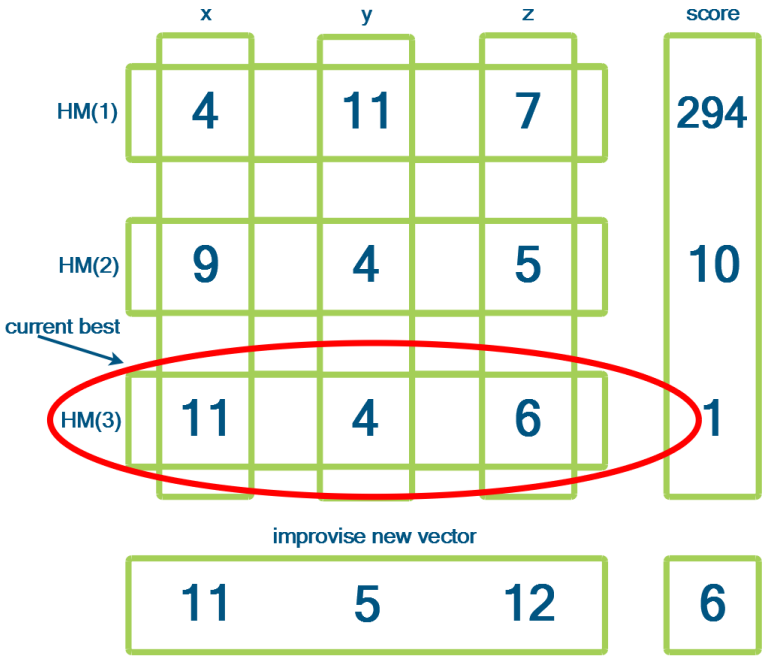

Figure 2.3: This is the HM together with a newly improvised solution vector. The new improvisation is better than the worst solution in the HM and will thus replace it (image taken from [23]).

The newly improvised vector is now evaluated and compared with the worst solution in the HM. Since the new improvisation, $\{11, 5, 12\}$, evaluates to a lower score, which is better since we are minimising, it replaces $\{4, 11, 7\}$ which is the worst vector in the HM. Notice that after the HM is updated that the worst solution is not necessarily the new improvisation but can be any vector in the HM. In this case the new worst is the second vector, $\{9, 4, 5\}$.

The final step of the algorithm is to check for convergence and either repeat the improvisation process if not converged or return the best solution vector in the HM as the optimum. Convergence detection and stopping criteria are typically problem

specific but the simplest method is to stop after a maximum number of improvisations. This method puts an upper bound on the amount of time harmony search spends on the problem. However, problem specific stopping criteria are usually more efficient and in some cases the solutions in the HM converge long before such arbitrary stopping criteria are met. In Section 2.5.2 we discuss convergence detection and advanced stopping criteria in more detail.

This simple example illustrates how harmony search optimises an objective function. We can see from the HM that even after one iteration the best solution is very close the optimum. However in practical problems the number of components are very seldom so small and often number in the hundreds. This combined with often wide upper and lower bounds on the component values dramatically increase the size of the search space and therefore the number of iterations required to find the optimum. As we will see in the rest of this thesis, harmony search can be adapted using various methods to keep the required iterations to a practical limit even when the search space becomes much too large for many other population based methods. However, the basic steps of harmony search, as illustrated in Figure 2.4 remain the same.

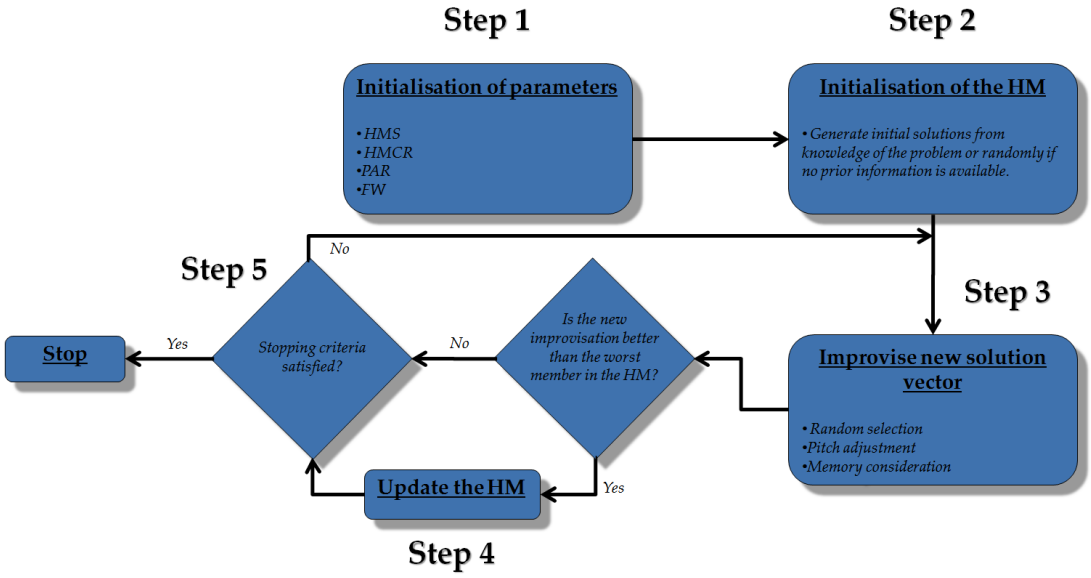

Figure 2.4: This diagram illustrates the main steps of harmony search.

2.3 Stochastic Derivative Using Harmony Search

The harmony search algorithm was designed to solve optimisation algorithms but it can also be used to analyse discrete functions in much the same way that differential calculus is used to analyse continuous functions. Geem proposed a novel stochastic derivative for discrete functions using harmony search [16]. This stochastic derivative gives a probability of selecting a specific discrete value for each component of the objective function that changes as the HM is updated through improvisation. The key insight here is that as the HM evolves with increasing improvisations, so the probability of choosing the optimum value for each component increases. The stochastic derivative can help identify local optima and gives insight into the shape of the search space which could lead to more efficient algorithms that are designed specifically for a certain search space.

In order to make the definition easier the pitch adjustment operator is simplified and made specific to discrete problems. This definition is slightly different from the one given in equation (2.1) and clamps the adjusted value to only one of two neighbouring values.

$$x'_{new} = x_{new} \pm m \ , \tag{2.2}$$

where x'_{new} is the pitch adjusted value of x_{new} and m is the step size to the neighbouring discrete value (similar to the fret width). Define the following.

$$P_{\text{random}} = 1 - \text{HMCR} \qquad \text{the probability of random selection} \tag{2.3}$$

$$P_{\text{memory}} = \text{HMCR}(1 - \text{PAR}) \qquad \text{the probability of memory consideration} \tag{2.4}$$

$$P_{\text{pitch}} = \text{HMCR}(\text{PAR}) \qquad \text{the probability of pitch adjustment} \tag{2.5}$$

$$K_i = |\{\text{LB}_i, \cdots, \text{UB}_i\}| \qquad \text{number of values for the } i\text{'th component} \tag{2.6}$$

The stochastic partial derivative of f based on harmony search is then defined as

$$\left.\frac{\partial f}{\partial x_i}\right|_{x_i=k} = \frac{1}{K_i} \cdot P_{\text{random}} + \frac{|\{x \in \text{HM}_i \mid x = k\}|}{\text{HMS}} \cdot P_{\text{memory}} + \frac{|\{x \in \text{HM}_i \mid x = k \pm m\}|}{\text{HMS}} \cdot P_{\text{pitch}} \ , \tag{2.7}$$

where HM_i is the i'th column of the HM and $|*|$ denotes set cardinality. The first term in the equation is the probability of the values k being chosen using random selection. The second and third terms are the probabilities of it being chosen using memory consideration and pitch adjustment.

I illustrate the stochastic derivative using the example from Section 2.2.1. The harmony search parameters are initialised as follows.

$$\text{HMS} = 3 \tag{2.8}$$
$$P_{\text{random}} = 0.1 \tag{2.9}$$
$$P_{\text{memory}} = 0.6 \tag{2.10}$$
$$P_{\text{pitch}} = 0.3 \tag{2.11}$$
$$m = 1 \tag{2.12}$$
$$x \in \{3,4,5,6,7,8,9,10,11,12,13,14\} \Rightarrow K_x = 12 \tag{2.13}$$
$$y \in \{3,4,5,6,7,8,9,10,11\} \Rightarrow K_y = 9 \tag{2.14}$$
$$z \in \{5,6,7,8,9,10,11,12\} \Rightarrow K_z = 8 \tag{2.15}$$

If we consider the HM directly after initialisation, as illustrated in Figure 2.2, the stochastic derivative for the optimal solution, $[11,4,7]$, is as follows.

$$HM = \begin{bmatrix} 4 & 11 & 7 & \vdots & 294 \\ 9 & 4 & 5 & \vdots & 10 \\ 11 & 4 & 6 & \vdots & 1 \end{bmatrix} \tag{2.16}$$
$$\Rightarrow \left.\frac{\partial f}{\partial x}\right|_{x=11} = \frac{1}{12}(0.1) + \frac{1}{3}(0.6) + \left[\frac{0}{3}(0.3)(0.5) + \frac{0}{3}(0.3)(0.5)\right] = 0.2083 \tag{2.17}$$
$$\left.\frac{\partial f}{\partial y}\right|_{y=4} = \frac{1}{9}(0.1) + \frac{2}{3}(0.6) + \left[\frac{0}{3}(0.3)(0.5) + \frac{0}{3}(0.3)(0.5)\right] = 0.411 \tag{2.18}$$
$$\left.\frac{\partial f}{\partial z}\right|_{z=7} = \frac{1}{8}(0.1) + \frac{1}{3}(0.6) + \left[\frac{0}{3}(0.3)(0.5) + \frac{1}{3}(0.3)(0.5)\right] = 0.2625 \tag{2.19}$$

To simplify this illustration pitch adjustment is split into two terms with the fist one representing the case where $x = k + m$ and the second one the case where $x = k - m$. We can already see from these results that the HM will quickly converge to the correct solution due the relatively large probability of selecting the optimum values even directly after initialisation. Note that this is not commonly the case but is rather

the result of using such a simple example with a severely limited search space centred around the solution.

Notice the effect that improvisation has on the stochastic derivative. Continuing with the example we calculate the stochastic derivative after the improvisation illustrated in Figure 2.3.

$$HM = \begin{bmatrix} 11 & 5 & 12 & \vdots & 6 \\ 9 & 4 & 5 & \vdots & 10 \\ 11 & 4 & 6 & \vdots & 1 \end{bmatrix} \tag{2.20}$$

$$\Rightarrow \left.\frac{\partial f}{\partial x}\right|_{x=11} = \frac{1}{12}(0.1) + \frac{2}{3}(0.6) + \left[\frac{0}{3}(0.3)(0.5) + \frac{0}{3}(0.3)(0.5)\right] = 0.4083 \tag{2.21}$$

$$\left.\frac{\partial f}{\partial y}\right|_{y=4} = \frac{1}{9}(0.1) + \frac{2}{3}(0.6) + \left[\frac{1}{3}(0.3)(0.5) + \frac{0}{3}(0.3)(0.5)\right] = 0.461 \tag{2.22}$$

$$\left.\frac{\partial f}{\partial z}\right|_{z=7} = \frac{1}{8}(0.1) + \frac{0}{3}(0.6) + \left[\frac{0}{3}(0.3)(0.5) + \frac{1}{3}(0.3)(0.5)\right] = 0.0625 \tag{2.23}$$

It is clear that the overall quality of the HM was improved even though probability of selecting $z = 7$ went down from 26.35% to 6.25%. After several more iterations we might see the following situation.

$$HM = \begin{bmatrix} 11 & 4 & 8 & \vdots & 1 \\ 12 & 4 & 7 & \vdots & 1 \\ 11 & 3 & 7 & \vdots & 1 \end{bmatrix} \tag{2.24}$$

$$\Rightarrow \left.\frac{\partial f}{\partial x}\right|_{x=11} = \frac{1}{12}(0.1) + \frac{2}{3}(0.6) + \left[\frac{0}{3}(0.3)(0.5) + \frac{1}{3}(0.3)(0.5)\right] = 0.4583 \tag{2.25}$$

$$\left.\frac{\partial f}{\partial y}\right|_{y=4} = \frac{1}{9}(0.1) + \frac{2}{3}(0.6) + \left[\frac{1}{3}(0.3)(0.5) + \frac{0}{3}(0.3)(0.5)\right] = 0.461 \tag{2.26}$$

$$\left.\frac{\partial f}{\partial z}\right|_{z=7} = \frac{1}{8}(0.1) + \frac{0}{3}(0.6) + \left[\frac{0}{3}(0.3)(0.5) + \frac{1}{3}(0.3)(0.5)\right] = 0.4625 \tag{2.27}$$

Now the HM is on the verge of finding the optimum solution and since all the solution vectors are neighbouring to the optimum solution, we know that the next time the HM is updated it will likely be with the optimum. This is also reflected in the probabilities of the stochastic derivative. Choosing the optimum value for each one of the components is now much more likely than any other values in the search space with an overall probability of $(0.4583)(0.461)(0.4625) = 0.0977$ to choose the optimum solution in the

next iteration. From this one can expect to see the algorithm reach the optimum solution within the next few iterations with a high probability.

The stochastic derivative tells us the probabilities of certain values being chosen in the next improvisation. This *stochastic derivative* acts different from a normal derivative in that the *rate-of-change* information normally associated with a derivative is now a probability. However, this probability tells us how likely (or quickly which leads to the derivative analogy) a specific value will show up in the HM [16]. This can give a designer of an algorithm to optimise a specific objective function valuable insight into the shape of the search space. It can also help designers to optimise the harmony search parameters by comparing the effect that improvisation has on a specific HM configuration for different parameter values. However, this tells us very little about harmony search's ability to thoroughly cover the entire search space of solutions and thus ensure that the global optimum is reached[1]. In the section that follows we investigate the exploratory power of harmony search and give a mathematical justification for harmony search's ability to thoroughly cover the entire search space. As we will see, effective coverage is dependent on the values of the harmony search parameters and they should be carefully chosen to minimise the risk of getting stuck in local optima.

2.4 Optimisation of the Exploratory Power of Harmony Search

The theoretical analysis of harmony search is an area that has not received as much attention as the many practical applications that harmony search has. Das et al. gives a simple mathematical analysis that provides insight into the exploratory power of harmony search [35, 36]. The focus of this analysis is the evolution of the population variance over successive generations as measured by the content of the HM. Through this an important conclusion is reached concerning the ability of harmony search to thoroughly cover the search space. This leads to a small modification in the algorithm that enhances the exploratory power of harmony search and it is shown to outperform

[1] Dr Z.W. Geem (in a private correspondence) points out that the search space can be more efficiently covered by generating more than HMS initial vectors and keeping only the best ones in the final initialisation.)

the original harmony search on many benchmark problems. The authors call their modified algorithm explorative harmony search (EHS).

The first step in the analysis is the derivation of a term describing the expected population variance. The analysis focusses on the pitch adjustment step since this is the primary source of diversity in the HM. When the HMCR is chosen high (this is standard practice, see [1, 2, 4, 5]) random selection rarely occurs and as the number of iterations increases its role in adding diversity decrease compared to the pitch adjustment step.

The aim is to show that the improvisation step causes the variance of the HM to grow quickly independent of how the HM is updated after improvisation. Since new improvisations are constructed component-by-component with each component considered independently, there is no loss of generality to constrain the analysis to single component problems. This means that the solution vectors in the HM are now scalars and the search space is now just a range of scalars between an upper and lower bound. To simplify the analysis an entire population of improvisations, $Y = \{y_1, y_2, \ldots, y_{\text{HMS}}\}$, is considered at each iteration instead the normal single improvisation. Each improvisation in Y is generated using Algorithm 2.1.

With these assumptions the variance of the initial population in the HM is as follows.

$$\text{Var}(\text{HM}) = \frac{1}{m}\sum_{l=1}^{m}(x_l - \overline{\text{HM}})^2 = \overline{\text{HM}^2} - \overline{\text{HM}}^2 \ , \tag{2.28}$$

where $\text{HM} = \{x_1, x_2, \ldots, x_m\}$ is the current population, $m = \text{HMS}$, and $\overline{\text{HM}}$ is the population mean. The following theorem (taken directly from [35]) describes the expected variance of the HM population after improvisation. For the proof of this theorem see [35].

Expected population variance theorem Let $\text{HM} = \{x_1, x_2, \ldots, x_m\}$ be the current population of the harmony memory and $Y = \{y_1, y_2, \ldots, y_m\}$ be the intermediate population obtained after the new harmony improvisation step. If we consider the allowable range for the component values in the HM to be $\{x_{min}, x_{max}\}$

where $x_{max} = a$ and $x_{min} = -a$ with $a \in \Re$, then

$$E(\text{Var}(Y)) = \frac{m-1}{m}\left[\text{HMCR} \cdot \text{Var}(HM) + \text{HMCR} \cdot (1 - \text{HMCR})\overline{\text{HM}}^2 + \frac{1}{3}\text{HMCR} \cdot \text{PAR} \cdot \text{FW}^2 + \frac{a^2}{3}(1 - \text{HMCR})\right] \quad (2.29)$$

From this theorem the following lemma, which describes the conditions under which one can expect exponential growth of the HM's variance, is derived. As before a full proof of this lemma can be found in [35].

Explorative harmony search lemma If the HMCR is chosen to be high (i.e. very near to 1) and the FW is chosen to be proportional to the standard deviation of the HM (i.e. $\text{FW} \propto \sigma(\text{HM}) = \sqrt{\text{Var}(\text{HM})}$), then the expected population variance (without updating the HM) can grow exponentially over iterations.

This lemma leads us to redefine the fret width as $\text{FW} = \sigma(\text{HM}) = k\sqrt{\text{Var}(\text{HM})}$ where k is a proportionality constant. Notice that if $\text{HMCR} \approx 1$ then equation (2.29) becomes

$$E(\text{Var}(Y)) \approx \frac{m-1}{m}\left[\text{HMCR} \cdot \text{Var}(HM) + \frac{1}{3}\text{HMCR} \cdot \text{PAR} \cdot \text{FW}^2\right]$$
$$\Rightarrow E(\text{Var}(Y)) \approx \frac{m-1}{m}\left[1 + \frac{1}{3}k^2 \cdot \text{PAR}\right]\text{Var}(\text{HM}) \cdot \text{HMCR} \quad (2.30)$$

From the explorative harmony search lemma we know to expect exponential growth from this term. From the proof (see [35]) we see that to ensure this growth the HMCR, PAR and k must be chosen so that $\left[1 + \frac{1}{3}k^2 \cdot \text{PAR}\right]\text{HMCR}$ is greater than 1.

The performance of EHS has been thoroughly tested and compared in [35] and was shown to be superior. EHS was tested on 15 well known unconstrained benchmark functions and consistently performed better than three other state-of-the-art harmony search variants namely, IHS, MHS and GHS.

This analysis provides us with a way to maximise the exploratory power of harmony search without diminishing its ability to take advantage of good solutions in the HM to accurately converge to the optimal solution. In most heuristic optimisation algorithms there is a trade-off between an algorithm's ability to thoroughly explore the search space and its ability to take advantage of neighbouring good solutions to accurately find the optimal solution. This trade-off is referred to as exploration versus exploitation.

To maximise exploration, the EHS algorithm maximised the variance of the HM. This combined with the exploitation of good solutions by replacing bad solutions in HM with good ones, makes EHS more robust to large search spaces with multiple local optima without significantly diminishing accuracy.

2.5 Implementation Details

As we saw in Section 2.2, harmony search is easy to implement and simple to understand. However, one can achieve significant performance gains by taking advantage of application specific features and adapting certain aspects of harmony search accordingly. In this section we focus on three areas in the implementation of harmony search that can be modified: HM initialisation, parameter optimisation and convergence detection. In all three these areas there are many possible implementations which are discussed and compared. For best results it is always recommended that the choice between different approaches be made based on what is known about the problem. For example, if it is known that the search space contains many local optima the parameters should be chosen to favour exploration above exploitation and convergence detection should be made more robust to guard against premature convergence. If it is known that good solutions are frequently found in particular areas of the search space, then this information can be used in the smart initialisation of the HM.

2.5.1 Initialisation Strategies

In the simple example of Section 2.2.1 we chose to initialise the HM randomly from the entire range of possible values for each component. This works well enough when the search space is small and real time performance is not required. However, there are much better ways to initialise the HM when the designer has some knowledge of the problem and knows where it is likely that good solutions can be found. As we will see in the sections dealing with applications, this is often the case.

Harmony search uses the solutions in the HM as the basis for new improvisations which means that the areas in the search space that harmony search explores first is directly influenced by the initial contents of the HM. As we saw in Section 2.3, the probability of choosing some values over others for the next improvisation is directly controlled by the prevalence of that number in the HM.

Therefore, if one knows nothing about the shape of the search space or where one could likely find good solutions, the best way to initialise the HM would be to maximise diversity by randomly choosing solutions from a uniform distribution over the entire search space. Each random solution then becomes a seed around which harmony search will try to find better solutions. This is essentially an unbiased search through the whole search space. The expectation is that as the solutions move closer together through improvisation and HM updates that at least one of them will pass through the global optimum. When the search space is very large this approach can take a very long time to find the global optima and is usually only used as a last resort. A pseudo code implementation of random initialisation is given in Algorithm 2.2.

input : The uninitialised HM
input : $d =$ the number of components in the solution vector
output: The initialised HM

```
1 foreach i ∈ [1, 2, ..., HMS] do  // loop through the vectors of the HM
      // s is an empty vector of length d + 1
2     s ← []
3     foreach j ∈ [1, d + 1] do            // loop through each component
4         s[j] ← r                                  // where r ∽ U(LB_i, UB_i)
5     end
6     s[d + 1] ← f(s)                      // f is the objective function
7
8     HM[i] ← s
9 end
```

Algorithm 2.2: Random initialisation

Another approach is to uniformly initialise evenly across the range of each decision variable. Instead of randomly initialising values are chosen evenly spaced across a decision variables range. However, when this approach is applied to each decision variable separately multidimensional clustering can result which defeats the purpose of maximising diversity. A better approach to non-random initialisation is to use multidimensional low discrepancy sequences to ensure that the search space is maximally covered without forming clusters [37]. This initialisation method is used in the AHS algorithm (see Section 2.6.4) and has shown to be effective in improving the convergence speed of harmony search based optimisers [14].

When a rough estimate of the optimal solution is available this can be used as a seed in the HM and leads to a much better initialisation strategy. In many applications the designer has some prior information from which a likely solution can be estimated. We will see examples of prior information being used in this way in Sections 3.4.1 and 4.3.6.

It is important to realise that prior information can only provide a rough estimate and that total reliance on this estimate will likely lead to getting trapped in a local optima. It would therefore be a mistake to initialise all the solution vectors in the HM to be equal to the estimated solution. Since there are then no diversity in the HM convergence to the optimum will be very slow if the estimate is far from the real solution. A better solution is to randomly distribute initial solutions around the rough estimate. We initialise the HM with random solutions from a normal distribution with the mean equal to the rough estimate and the variance proportional to the confidence we have in the estimate. Pseudo code for this technique, called seeded initialisation, is given in Algorithm 2.3. Notice that the real difference between random and seeded initialisation is only the distribution of r and the origin of that distribution's parameters.

```
   input  : The uninitialised HM and the estimate e
   input  : d = the number of components in the solution vector
   input  : σ = error variance of e
   output: The initialised HM
 1 foreach i ∈ [1, 2, ..., HMS] do  // loop through the vectors of the HM
       // s is an empty vector of length d + 1
 2     s ← []
 3     foreach j ∈ [1, d + 1] do            // loop through each component
           // s[j] is normally distributed around e[j]
           // σ is proportional to the confidence in e
 4         s[j] ← r                                  // where r ∽ N(e[j], σ)
 5     end
 6     s[d + 1] ← f(s)                      // f is the objective function
 7     HM[i] ← s
 8 end
```

Algorithm 2.3: Seeded initialisation

Seeded initialisation has the advantage of taking advantage of prior information without risking slow convergence should the estimate be far from the true optimum solution. As we will see in Section 3.4, seeded initialisation can also be combined with

premature convergence detection to force harmony search to intentionally get stuck in a specific local optima. This is useful when the objective function does not perfectly describe the best solution to the problem and local optima may turn out to be the best solution.

In some situations the components of the solution vector can be split into distinct parts of the search space. Prior information might be available for some of the components but not for all. A hybrid initialisation scheme can be used to initialise some of the components using the available prior information while other components, of which nothing is known, are randomly initialised. This ensures that all information that can possibly speed up convergence are used. Prior information can either be used as the centre of a random distribution like we saw in seeded initialisation or in specialised cases some components are all set equal to the same prior. Examples of hybrid initialisation used in this specialised way is discussed in Section 5.3 and Section 5.4. Pseudo code for hybrid initialisation is shown in Algorithm 2.4.

input : The uninitialised HM and the prior p
input : $d =$ the number of components in the solution vector
output: The initialised HM

1 **foreach** $i \in [1, 2, \ldots, HMS]$ **do** // loop through the vectors of the HM
  // s is an empty vector of length $d + 1$
2 $\mathbf{s} \leftarrow []$
3 **foreach** $j \in [1, d + 1]$ **do** // loop through each component
4 **if** $j = d + 1$ **then** // evaluate s
5 $\mathbf{s}[j] \leftarrow f(\mathbf{s})$ // f is the objective function
6 **else if** $p[j] \neq \#$ **then** // p is a vector of prior values.
  // $p[j] = \#$ indicates no prior value is available
  // p can also be used like e from Algorithm 2.3
7 $\mathbf{s}[j] \leftarrow p[j]$
8 **else** // build s
9 $\mathbf{s}[j] \leftarrow r$ // where $r \backsim U(\mathtt{LB}_i, \mathtt{UB}_i)$
10 **end**
11 **end**
12 $\mathrm{HM}[i] \leftarrow \mathbf{s}$
13 **end**

Algorithm 2.4: Hybrid initialisation

Ultimately the choice of initialisation strategy should be based on the information

available that can possibly guide harmony search to likely solutions. Random initialisation is only recommended when the search space is small and nothing is known about likely solutions prior to optimisation. Seeded initialisation should be used when an estimate can be made of a likely solution. When confidence in the estimate is high σ can be set small to ensure quick convergence to the solution close to that estimate. When confidence is low, σ should be chosen larger to ensure sufficient diversity to escape the estimate's basin of attraction, should it be the wrong solution. Hybrid initialisation should be used when a complete estimate is not available but some of the components can be estimated from prior information.

2.5.2 Detecting Convergence

Like the initialisation strategies, convergence detection should also be designed specifically for the application. In applications that do not have strict run-time constraints efficient convergence detection is not of great concern and simply halting the optimisation after a maximum number of iterations were reached is usually sufficient. However, in real-time applications or in applications where a timely execution is preferable, convergence detection is very important.

As mentioned, the simplest approach is to count the number of iterations and stop when some maximum has been reached. This approach has the advantage of not adding any extra computational overhead and being the easiest to implement. The run-time versus accuracy trade-off can be accurately controlled by setting the maximum number of iterations low for quick execution and high for slower execution but potentially better accuracy. The disadvantage of this approach is that many iterations are likely being wasted since the optimiser will only halt when the maximum iterations have been reached even when the optimal solution has stopped improving.

In time critical applications it is important to detect when the optimiser will likely not be able to improve on the current solution any further. Optimisation should then immediately be stopped to prevent wasted iterations. A simple way to minimise wasted idle iterations is to count the number of consecutive idle iterations and halt optimisation when a set number of consecutive idle iterations have been counted. An idle iteration is identified as an iteration in which the HM is not updated. In other words, the newly improvised solution is worse than the worst solution in the HM and is therefore discarded without affecting the HM.

The danger of this approach lies in the assumption that multiple consecutive idle iterations imply that no further progress will be made. As is often the case, the majority of candidate solutions in the HM may be in the basin of attraction of a local optimum that is not the global optimum. The more solutions close to the local optimum the longer it will take to escape the basin of attraction and generate an improvisation that is good enough to update HM. It might take many idle iterations before a new local optimum is found starting a new series of updates to the HM. The number of idle iterations to allow before halting should therefore be carefully chosen specific to the search space to prevent premature convergence to suboptimal solutions.

Another approach that does not rely on counting idle iterations is to measure the diversity in the quality of the solutions directly. Eventually, all the candidate solutions will converge to the same basin of attraction and gradually come together until they all equal the same optimum. By measuring the Euclidean distance between the best and the worst candidate in the HM we can get a measure of how closely the candidates have converged towards the same solution. When this distance is smaller than some set amount, ϵ, the HM is considered converged and the optimiser halts. Normally one chooses ϵ proportional to the desired accuracy, setting ϵ equal to the largest error one is willing to make. However, depending on the size of the HM this may take very long for all the candidates in the HM to move close enough together and a larger ϵ is usually more practical. Given a large enough initial diversity in the HM, this is usually the safest way of convergence detection but some iterations are still wasted since it is unnecessary to wait for all the candidates in the HM to converge when only one equal to the optimum is required. Dr Z.W. Geem suggests (in a private correspondence) that the FW can be decreased when convergence is first suspected in an attempt to inject diversity into the HM without immediately halting convergence. This could allow for more sensitive convergence detection resulting in less wasted iterations while minimising the risk of premature convergence.

A good trade-off between the advantages of all the previously mentioned methods is to combine them into a set of tests. If any of the tests indicate convergence, the optimiser halts and the best solution in the HM is considered the optimal solution. If ϵ and the maximum number of idle iterations are chosen carefully this method is very efficient in quickly detecting convergence with the minimum of iterations wasted. As we will see in Section 3.4, it is even possible to do real-time video tracking using this

method for convergence detection. A pseudo code implementation of the combined convergence tests is shown in Algorithm 2.5.

```
   input  : A non-converged HM
   input  : g = the current number of iterations
   input  : g_idle = the current number of iterations
   input  : G_max = the maximum iterations allowed
   input  : G_idleMax = the maximum idle iterations allowed
   output: The convergence status of the HM
 1 if g_idle > G_idleMax then                    // check idle iterations
 2 |  HALT optimisation
 3 // HM_best and HM_worst are the best and worst members of the HM
 4 else if |HM_best − HM_worst| < ε then              // check diversity
 5 |  HALT optimisation
 6 else if g > G_max then                        // check max generations
 7 |  HALT optimisation
 8 else
 9 |  CONTINUE with new improvisation
10 end
```

Algorithm 2.5: Combined convergence tests

The regular testing for convergence using diversity measurements and the counting of idle iterations in real-time applications has not been applied to harmony search before and is a novel contribution of this research. I introduced this method in [24] with further details in [23].

2.5.3 Parameter Optimisation and Sensitivity

The aim of this section is to investigate the effect that the choice of the HMS and the HMCR has on harmony search. In choosing the values of these parameters one tries to find a balance between the accuracy of the final solution and minimising the number of iterations required to reach it. These are conflicting factors in many optimisation algorithms and are mainly controlled by the HMS, HMCR and PAR in harmony search. We wish to optimise these parameters in such a way that the desired accuracy can be consistently achieved using the minimum amount of iterations. Therefore, an analysis of the sensitivity of the effect these parameters have on the accuracy and performance of harmony search is required. Overly sensitive parameter settings are undesirable as this

forces the designer to fine tune parameters with each new application of the algorithm.

Analysis of the PAR is deferred until Section 3.4.5. As we will see in Section 2.6 the PAR parameter can be used more effectively when its value is not constant but rather varied depending on the current iteration. The PAR is also often completely removed from modern harmony search variants as pitch adjustment is often replaced by alternate methods. For this reason I will rather analyse the PAR in Section 3.4.5 as part of my analysis of the Harmony Filter algorithm.

The sensitivity analysis of the HMCR and the HMS in this section is a novel contribution of this research and was first published in [23]. A more thorough analysis of the standard harmony search algorithm was later published by Greblicki te al. [38]. However, the results of this section are taken from the research on the Harmony Filter algorithm in [23].

The analysis starts with the HMS. A small HMS means fewer candidates are considered during improvisation which means that the HM diversity will decrease more quickly. When the search space is not very large and confidence in the initialisation is high, this will likely result in quick convergence to the optimal solution. A higher HMS will cause slower convergence but minimises the risk of getting stuck in local optima by keeping the HM diversity larger for longer.

In the first test harmony search is adapted for tracking a visual target through a video sequence. The resulting algorithm, called the Harmony Filter (HF) is the subject of Section 3.4. The hybrid initialisation scheme of Algorithm 2.4 is used for initialisation and the combined convergence tests of Algorithm 2.5 are used as stopping criteria.

I vary the HMS between 5 and 30 and use the HF to calculate the position of the target. Since the HF continues to improvise new solutions until convergence to the desired accuracy is achieved, I concentrate the analysis on the number of iterations taken to achieve convergence. The example video contains 480 separate frames and the HF is used to calculate the target's position in each frame. I then count the number of iterations until convergence in each frame and average them over all frames for each value of the HMS. The averages for each HMS is shown in Figure 2.5.

From this analysis we find that the number of iterations slowly increase with the HMS and that the best performance is found at HMS = 7. At this value the HM converged to an acceptable accuracy within an average of 102 iterations. Notice also that the graph tends to become more noisy as the HMS increases. As the HMS increases

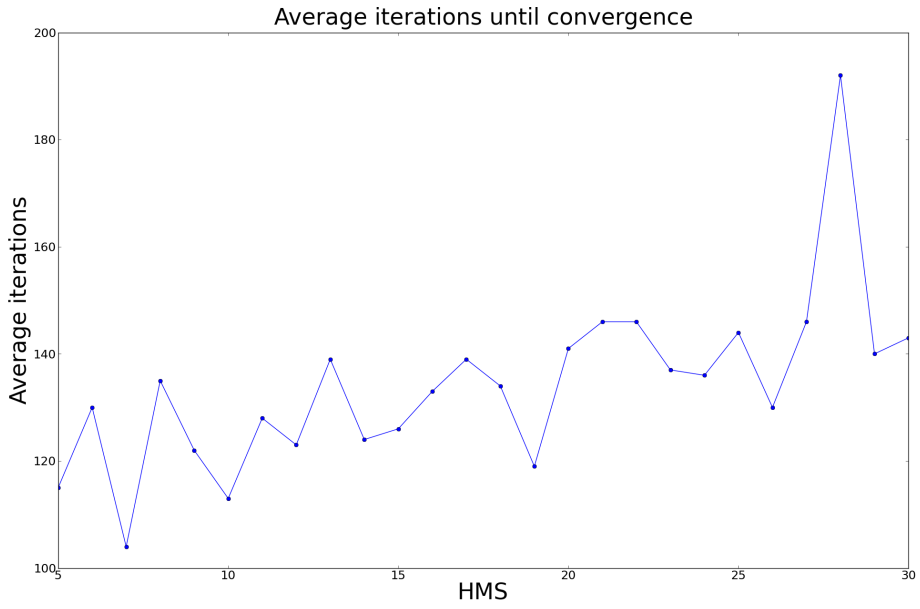

Figure 2.5: The graph shows the average number of iterations until convergence for various values of HMS. The minimum number of iterations are found at HMS = 7.

HS increasingly tends to resemble a randomised search leading to the higher peaks in the graph. It is also interesting to see that the algorithm is not overly sensitive to the choice of HMS as most of the values lie below 160 iterations with the worst score still below 200. Anything below 170 iterations was still considered acceptable performance for real-time tracking of the target. It was found that low HMS values give superior performance in applications with high confidence in the initialisation of the HM. These results agree with the results of Omran et al. that reports best results for HMS between 2 and 10 [17].

However, these results do not agree with those of Greblicki et al. that reports best results with a large HMS of 150 [38]. In their experiment HS is used to solve a well known optimisation problem called the binary knapsack problem [39]. The search space of this problem is much larger than in the example problem used in this section and nothing is known about the search space so random initialisation is used. The focus of these tests was accuracy instead of speed so no convergence tests were done. Instead the number of iterations was made constant at 5000 with optimisation only stopping

when the full number of iterations has been reached. It is therefore not surprising that the best results were achieved with a much larger HM. With a larger search space HS benefits from the increased diversity that is easier maintained with a large HM. The constantly high number of iterations available for the HM to converge to the best solution also favours a larger HMS as the slow convergence of a higher HMS is not penalised. However, the sensitivity of the HMS to the accuracy agrees with the results of this section which indicate that the speed is insensitive in the value of the HMS.

I conclude that both the speed and accuracy are insensitive to small changes of the HMS and that the HMS can be chosen according to the values recommended in HS literature [1, 5] without affecting the performance significantly. A good guideline is to choose the HMS proportional to the size of the search space and the amount of iterations that are available. The HMS can be significantly decreased if there is high confidence in the initialisation.

The second analysis focuses on the HMCR parameter and the same video tracking example used in the analysis of the HMS is again used here. The HMCR varies between 0 and 1 and the recommended practice is to set it high to favour memory consideration over random selection [1, 4]. The HMCR was varied between 0.5 and 0.99 and then the average number of iterations until convergence was calculated over all 480 frames. The results are shown in Figure 2.6.

We see slightly more variation from the HMCR showing that the performance is a slightly more sensitive to the HMCR as it was too the HMS. However, the performance is still acceptable for each value of the HMCR leading one again to conclude that the performance of harmony search is not sensitive to slight parameter changes.

The best performance was achieved at HMCR = 0.95 which agrees with the recommended practice of keeping the HMCR high. It also agrees with the findings of Greblicki that achieved the best results at the highest value he tested which was HMCR = 0.82. It is also interesting to note that = 0.95 is the exact same value that Mahdavi et al. considered optimal for the HMCR in their experiments with the improved harmony search (IHS) algorithm [13]. As the analysis in Section 2.4 showed, keeping the HMCR ≈ 1 also has the added advantage of leading to exponential increase in the HM diversity when the FW is chosen correctly. However, choosing the HMCR = 1 is not recommended especially when the FW is chosen small as this makes escaping from local optima much more difficult. In some problems where it is known that the search space

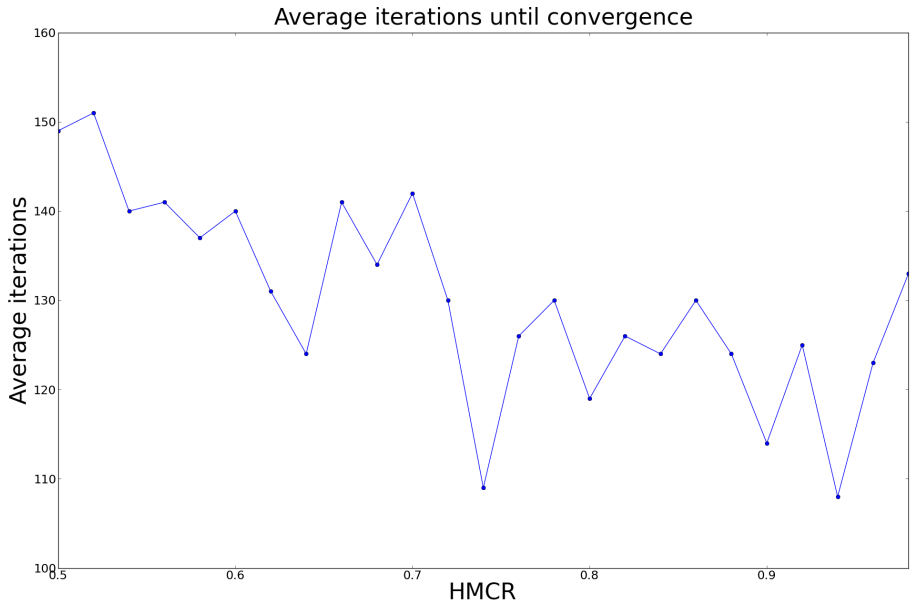

Figure 2.6: The graph shows the average number of iterations until convergence for various values of HMCR. The minimum number of iterations are found at HMCR = 0.95.

has many local optima with large basins of attraction it is often better to choose the HMCR slightly lower than usually recommended. In Section 5.4 we see an example of such a problem and it is found that for that special case HMCR = 0.85 gives the best results.

An important observation here is that this analysis on harmony search parameters was made specifically for use in the harmony filter (see Section 3.4). The harmony filter is a specific application of harmony search to visual tracking and the conclusions of this section is therefore somewhat specific to one application of harmony search. However, in practice, I found that one can generalize the conclusions of this section to most applications of harmony search. A notable exception to this statement is when the HM is separated into parts that are improvised differently (see Section 5.4), in which case a different set of parameters work better. It is for this reason and for the sake of future reference in the rest of the thesis that this analysis was discussed in this chapter and not deferred to the chapter on the harmony filter.

2.6 Improvements and Adaptations

Since the original introduction of harmony search in 2001 several researchers have proposed improvements and variants that enhance the performance in some way. In this section I introduce several harmony search variants and investigate the methods used to achieve these enhancements. The harmony search variants introduced in this section are all designed to be generic optimisation algorithms like the original harmony search and enhancements were not made to be problem specific. In Sections 3, 4 and 5 I will use similar methods to create problem specific variants of harmony search that are designed to fit specific purposes.

2.6.1 Harmony Search with Ensemble Consideration

One of the first generic improvements to harmony search followed naturally from the musical analogy of harmony search. The original creator of harmony search, Dr. Geem, realised that just as some instruments in a band share a special relationship and tend to track each other's behaviour, so too can some of the components in a solution vector. This realisation led him to add an operation to harmony search, called ensemble consideration, that takes advantage of correlations that might exist between different components [12].

In Figure 2.7 we see a musical example showing the score of three instruments. Simply by noting visually that the middle instrument always plays one tone higher than the top instrument we can see a strong relationship between the top two instruments that is not visible in the bottom one. It is reasonable to assume that the top instruments will continue to follow this trend enabling us to predict with a high probability what note will be played by one of the instruments based on what the other is playing.

Ensemble consideration is incorporated into harmony search by adding a step to the improvisation process. After improvisation of x_{new} (see Algorithm 2.1) each of its components are considered for ensemble consideration using a newly introduced parameter called the ensemble consideration rate (ECR). Like the HMCR the ECR is a parameter between 0 and 1 and controls the frequency of the ensemble consideration operator. For each component a random number between 0 and 1 is generated and if it is less than the ERC it is modified using ensemble consideration, otherwise it is left unmodified.

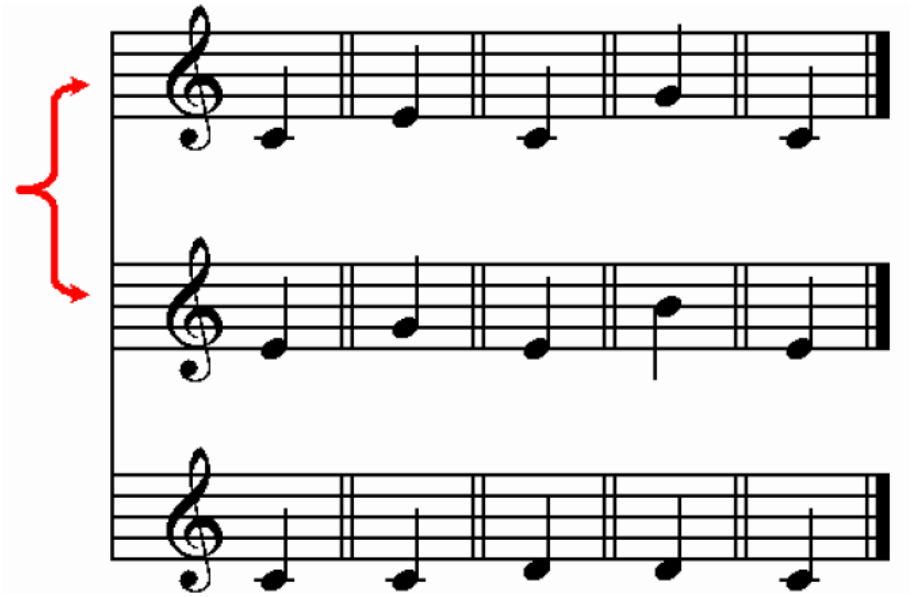

Figure 2.7: The musical score of these three instruments show a strong relationship between the top two instruments that is not visible in the bottom instrument. One can say that the top two instruments are strongly correlated (image from [12]).

Ensemble consideration modifies the component value based on a relationship function with the component that is most highly correlated with the component to be modified. The new component value is set with the following equation.

$$x'_i \leftarrow fn(x'_j) \text{ where } j \text{ is given by } \max_{i \neq j} \left\{ corr(\mathbf{x^i}, \mathbf{x^j})^2 \right\} \tag{2.31}$$

where $\mathbf{x^i} = [x_i^1, x_i^2, \ldots, x_i^{\text{HMS}}]$ and $\mathbf{x^j} = [x_j^1, x_j^2, \ldots, x_j^{\text{HMS}}]$. The relationship function, fn, finds the most frequent value of x_i from the pairs of $\mathbf{x^i}$ and $\mathbf{x^j}$ in the HM and in x_{new} where $x_j = x'_j$.

The correlation operator used, $corr(\mathbf{x^i}, \mathbf{x^j})$, is also known as the Pearson product-moment correlation coefficient [40] defined as

$$r_{\mathbf{x^i},\mathbf{x^j}} = \frac{\sum_{k=0}^{M} \mathbf{x^i}_k \mathbf{x^j}_k - M\overline{\mathbf{x^i}}\ \overline{\mathbf{x^j}}}{(M-1)\sigma_{\mathbf{x^i}}\sigma_{\mathbf{x^j}}} \tag{2.32}$$

where $\sigma_{\mathbf{x^i}}$ and $\sigma_{\mathbf{x^j}}$ is the standard deviation of $\mathbf{x^i}$ and $\mathbf{x^j}$. HS uses r^2 as the correlation measure and it is also known as the *determination coefficient* and can be calculated

from its expanded form defined (see [40]) as

$$r^2_{\mathbf{x^i},\mathbf{x^j}} = \frac{\left(M\sum_{k=0}^{M}\mathbf{x^i}_k\mathbf{x^j}_k - \sum_{k=0}^{M}\mathbf{x^i}_k\sum_{k=0}^{M}\mathbf{x^j}_k\right)^2}{\left(M\sum_{k=0}^{M}\mathbf{x^i}^2_k - \left(\sum_{k=0}^{M}\mathbf{x^i}_k\right)^2\right)\left(M\sum_{k=0}^{M}\mathbf{x^j}^2_k - \left(\sum_{k=0}^{M}\mathbf{x^j}_k\right)^2\right)} \tag{2.33}$$

Pseudo code for the improvisation operation with ensemble consideration is shown in Algorithm 2.6. On line 14 the determination coefficient is calculated using equation (2.33) which is then used to determine $\mathbf{x^j}$. On lines 15 and 16 two sets are constructed for the calculation of the relationship function, fn. Λ_i contains all the indices of $\mathbf{x^j}$ that correspond to values equalling x'_j from x_{new}. These indices are then used to construct the set Ψ_i that contains values from $\mathbf{x^i}$ that correspond to those indices. Finally, the number a times a certain value appears in Ψ_i is counted (note that $|\cdot|$ denotes set cardinality) and the value that appears most often is used as the new component value.

Dr Geem reports improved results from using ensemble consideration and has since then made it an optional addition to the standard harmony search algorithm [5, 12]. The ECR parameter should be kept much lower than the HMCR with the best results reported with ECR = 0.01.

2.6.2 Improved Harmony Search

Mahdavi et al. noticed that the PAR and FW parameters are important to the fine tuning of optimised solution vectors and suggested a method of adapting PAR and FW to the relative progress of the optimiser instead of keeping these parameters constant through all iterations [13]. They argue that the PAR and FW parameters are often difficult to choose exactly and that a range of values should instead be used. They call this approach the improved harmony search algorithm (IHS).

According to their theory, large values for the FW parameter are needed in the early iterations to maintain diversity and avoid local optima, while smaller values should be used during the final iterations when fine tuning of already good solution vectors should be the focus. Conversely, the PAR should be small during the initial iterations when the focus is on exploring the search space and large during the final iterations when it should encourage fine tuning of solutions.

input : The initialised HM
output: The new improvisation $x_{new} = [x'_0, ..., x'_M]$

foreach $i \in [1, M]$ **do**
  if $U(0,1) > HMCR$ **then** // random selection
    $x'_i \leftarrow \mathrm{LB}_i + r \times (\mathrm{UB}_i - \mathrm{LB}_i)$ // where $r \backsim U(0,1)$
  else // memory consideration
    $x'_i \leftarrow x^j_i$ // where $j \backsim U(1, \mathrm{HMS})$
    if $U(0,1) \leq PAR$ **then** // pitch adjustment
      $x'_i \leftarrow x'_i + r \times \mathrm{FW}$ // where $r \backsim U(-1,1)$
    end
  end
  $x_{new}[i] \leftarrow x'_i$
end
foreach $i \in [1, M]$ **do**
  if $U(0,1) < ECR$ **then** // ensemble consideration
    $j \leftarrow \max_{i \neq j} \left\{ r^2_{\mathbf{x^i}, \mathbf{x^j}} \right\}$ // $r^2_{\mathbf{x^i}, \mathbf{x^j}}$ as defined in equation (2.33)
    // $\mathbf{x^i}$ and $\mathbf{x^j}$ is used like in equation (2.31)
    $\Lambda_i \leftarrow \{k \mid x_k = x'_j \ \forall \ x_k \in \mathbf{x^j}\}$ // indices from $\mathbf{x^j}$
    $\Psi_i \leftarrow \{p \mid p = \mathbf{x^i}_k \ \forall \ k \in \Lambda_i\}$ // corresponding $\mathbf{x^i}$ values
    $x_{new}[i] \leftarrow \alpha$ where α maximises $\max_{\alpha \in \Psi_i} |\{p \mid p = \alpha \ \forall \ p \in \Psi_i\}|$
  end
end

Algorithm 2.6: The improvisation step with ensemble consideration

It is suggested that the PAR and FW be calculated based on the elapsed iterations using the following equations.

$$PAR(i) = PAR_{min} + \frac{PAR_{max} - PAR_{min}}{NI - 1} \cdot (i - 1) \tag{2.34}$$

$$FW(i) = FW_{max} e^{(k \cdot i)} \tag{2.35}$$

$$\text{with } k = \frac{\ln(\frac{FW_{min}}{FW_{max}})}{NI} \ , \tag{2.36}$$

where PAR_{min}, PAR_{max} are the minimum and maximum possible values of the PAR and FW_{min}, FW_{max} are the same with respect to the BW parameter. NI is the total number of iterations and i is the iteration index. This causes the PAR to climb linearly from the minimum value at the start of the process to its maximum when the total

number of iterations have been reached. The BW starts off at its maximum value and then falls exponentially until it reaches its minimum at the end of the optimisation.

Mahdavi et al. reports greatly improved results compared to original harmony search but other research suggests that this approach is not better in all situations and contradictory results have been reported [14, 17, 41]. Even the original authors of IHS later reported better results with a constant, relatively low valued, PAR [17].

The main purpose of IHS was to remedy the weakness in HS of a fixed PAR and BW that is often difficult to choose. IHS succeeded in addressing this limitation but the method used seems questionable.

Taherinejad and Wang et al. argue that the PAR should be large during the initial iterations and gradually reduced similar to the approach used in simulated annealing (high temperature gradually decreasing with increasing iterations) [14, 41]. Taherinejad suggests that the PAR should be updated as follows.

$$PAR(i) = PAR_{max} - \frac{PAR_{max} - PAR_{min}}{NI - 1} \cdot (i - 1) \tag{2.37}$$

This is clearly opposite to the approach used in IHS but the reported results indicate that this is a more reliable approach that usually leads to better results. Wang et al. also linearly decreases the PAR but uses a very different approach to the improvisation step that will be investigated further in Section 2.6.4.

2.6.3 Global Best Harmony Search

The global best harmony search algorithm (GHS) was inspired by the swarm intelligence concept used in the particle swarm optimisation algorithm (PSO) [34]. A social element is added to the pitch adjustment step by moving the component values towards the value of the best member in the HM. This method has the advantage of not requiring the FW parameter at all alleviating the problem of tuning the FW *a priori* [17]. It is also able to optimise both discrete and continuous problems equally well. GHS is identical to standard HS in every step except for the pitch adjustment operation that is modified as follows.

$$x_i' \leftarrow x_i^{best} \tag{2.38}$$

where x_i^{best} is the ith component value of the best member in the HM. Referring to the standard improvisation of Algorithm 2.1 this equation would replace the equation in line 7 to make the improvisation step used in GHS.

Results from GHS were compared with those of standard HS and IHS using 10 well known benchmark optimisation problems. It was shown that the average performance of GHS over 50,000 evaluations was significantly better than both HS and IHS for all but one example.

The effect that the HMS, HMCR and PAR has on GHS's performance was also investigated. As previously also shown, It was found that high values for the HMCR (≥ 0.9) gave the best results in high dimensional problems. A very interesting result from this study was that a small constant value for the PAR usually outperforms an implementation where the PAR is linearly increased as suggested in Section 2.6.2. This result was shown using many different optimisation problems with a range of configurations and number of dimensions, which makes a convincing argument for keeping the PAR small and constant instead of a gradual increase.

2.6.4 Self-adaptive Harmony Search

The self-adaptive harmony search algorithm (SAHS) is another highly successful approach to alleviating the problem of how to choose the PAR and FW parameters in standard harmony search [14]. As suggested in Section 2.6.2, SAHS decreases the PAR linearly instead of the linear increase originally suggested in the IHS algorithm.

Like the GHS algorithm discussed in the previous section, SAHS does not use the FW parameter at all but instead replaces the pitch adjustment operator with an operation that modifies the new improvisation based on the values in the HM. However, this is where the similarities end.

The authors of SAHS argue that harmony search should not be classified with the other population based algorithms like genetic algorithms and particle swarm optimisation since harmony search only evaluates one potential solution at a time similar to tabu search [42] and simulated annealing [32]. They argue that using the population global best like GHS leads to premature convergence since the whole population is not updated during each iteration.

They suggest that the pitch adjustment step be replaced with something that can better use the harmony memory's past experiences without risking premature convergence. The pitch adjustment step is replaced by the following two equations.

$$x'_i \leftarrow x'_i + \left[\max(\mathrm{HM}^i) - x'_i\right] \cdot r \tag{2.39}$$

$$x'_i \leftarrow x'_i - \left[x'_i - \min(\mathrm{HM}^i)\right] \cdot r \ , \tag{2.40}$$

where $\max(\mathrm{HM}^i)$ and $\min(\mathrm{HM}^i)$ are the largest and smallest value found in the HM for the ith component and $r \backsim U(0,1)$. Each time pitch adjustment is performed the improvisation is updated by randomly applying, with equal probability, one of these two equations. This approach causes progressively smaller changes to be made to the new improvisation as $\max(\mathrm{HM}^i)$ and $\min(\mathrm{HM}^i)$ converge closer together with increasing iterations. Therefore, pitch adjustment starts off as a rough operator making larger changes to favour exploration, and then becomes a fine operator favouring exploitation as the optimiser converges closer to the optimum.

The performance of SAHS was thoroughly compared with that of standard HS, IHS and GBH using four standard optimisation benchmark functions, namely the Sphere, Rosenbrock, Ackley and Griewank equations. All test runs were done in both 30 and 100 dimensions over a range of parameter values. SAHS performed significantly better than all compared methods under a 99% confidence interval t-test in all test functions [14].

The results from this study gives further evidence that linearly increasing the PAR with increasing iterations does not improve the performance of HS, or at least does not improve performance unless combined with other modifications. Notice also that the terms that replace the FW parameter in standard HS, namely $\left[\max(\mathrm{HM}^i) - x'_i\right]$ and $\left[x'_i - \min(\mathrm{HM}^i)\right]$ vary in proportion to the standard deviation of the harmony memory. One could therefore say that a theoretical basis for the improved results of SAHS is found in the study of Section 2.4 that showed that the exploratory power of harmony search can be maximised if the FW is kept proportional to the harmony memory's standard deviation. However, the advantage of this approach is in the computational cost. It is computationally cheaper to maintain $\max(\mathrm{HM}^i)$ and $\min(\mathrm{HM}^i)$ than it is to calculate $\sqrt{Var(\mathrm{HM})}$ each time pitch adjustment is performed.

2.6.5 Dynamic Local Best Harmony Search

Another variation of harmony search inspired by PSO was introduced by Pan et al. [43]. Their approach, called Dynamic Local Best Harmony Search (DLHS), attempts to balance fast convergence and a large population diversity by dynamically dividing the HM into subpopulations. Each subpopulation is optimised individually using a local PSO algorithm. To maintain diversity, subpopulations are regrouped frequently using a regrouping schedule.

The optimisation process is divided into two phases. During the first phase the HM is divided into m smaller sub-HM's. Each sub-HM is then allowed to converge independently using the GHS scheme discussed in Section 2.6.3. In order to maintain diversity, the sub-HM's are randomly regrouped every R iterations. Then each new sub-HM restarts the search process until the first phase completes. In this way information is allowed to be shared among sub-HM's while simultaneously maintaining the diversity necessary to minimise getting stuck in local optima.

In the final phase DLHS focusses on local exploitation by using the best N solution vectors to form a single HM (in the original article the best 3 are used). This new HM is then optimised until the convergence criteria is reached.

A self adapting strategy is used to dynamically update the HMCR and PAR parameters. It uses two sets of possible HMCR and PAR parameters called the parameter set list (PSL) and the winning parameter set list (WPSL). First the PSL is generated by randomly filling it with HMCR and PAR values (HMCR $\in \{0.9, 1\}$ and PAR $\in \{0, 1\}$). During each improvisation values for the HMCR and the PAR is chosen from the PSL. If these values resulted in an improvisation that updated the HM they get transferred to the WPSL. Once the PSL is empty is gets refilled by randomly selecting parameter sets from the WPSL (75%) or randomly generated parameter sets (25%). In this way the best values for the HMCR and the PAR is gradually learned. The size of the PSL was experimentally set at 200 sets of HMCR and PAR values.

In order to balance between exploration of the search space and local exploitation of solutions, the FW is dynamically decreased in a similar way to that used in IHS (see Section 2.6.2) and SAHS (see Section 2.6.4) algorithms. However, instead of an

exponential decrease, DLHS uses a linear decrease defined as follows.

$$FW(i) = \begin{cases} FW_{max} - \frac{FW_{max} - FW_{min}}{NI} \cdot 2NI & \text{if } i < \frac{NI}{2} \\ FW_{min} & \text{otherwise} \end{cases} \tag{2.41}$$

where FW_{max} and FW_{min} are the maximum and minimum values for the FW, NI is the maximum number of iterations and i is the iteration index. For further implementation details see [43].

The performance of DLHS was compared with that of standard HS, IHS, GHS and MHS based on 16 benchmark functions. They were implemented in both 30 and 50 dimensions and are allowed to run for 50,000 iterations. The average error of 30 repetitions are then compared between the five algorithms. It was shown that DLHS performs significantly better on most benchmark functions for both the 30 and 50 dimensional cases. Statistical significance was calculated using the two-sided paired t-test with a 5% significance level.

2.7 Hybrid Approaches

In the last section we investigated various ways that researchers have adapted harmony search to address some of its shortcomings or to improve its performance. Instead of adapting harmony search directly one could also combine it with other proven methods. The usual strategy is to replace one or more steps of harmony search with another algorithm that is expected to perform that operation more accurately or faster. In particular, many researchers noticed that harmony search excels at efficiently covering a large search space but takes many iterations to converge to an accurate solution once the optimal solution's basin of attraction has been found. Harmony search thus tends to favour exploration over exploitation and many hybrid algorithms are based on replacing the improvisation step with an algorithm that is known to be good at exploiting local solutions to quickly converge to the optimum.

Another popular method of designing hybrid algorithms is to add another optimisation step as a pre- or post-processing step to harmony search. This is done to improve the quality of the HM and either serves to initialise it (preprocessing step) or to refine it after improvisation (postprocessing step). This section provides an overview of several

hybrid harmony search algorithms that aim to improve harmony search by combining it with a diverse range of traditional optimisation algorithms including sequential quadratic programming, particle swarm intelligence, simulated annealing and others.

2.7.1 Harmony Search + Simplex

The Nelder-Mead Simplex algorithm (NM-SA) is an evolutionary algorithm designed for unconstrained optimisation [44]. It is simple to implement and does not require derivative information from the function to be optimised. Jang et al. proposed the simplex-harmony search (SHS) algorithm as a hybrid between standard HS and the NM-SA algorithm [45]. The strategy is to improve the accuracy of HS by updating the harmony using a NM-SA step after each improvisation.

SHS starts by randomly initialising the HM with M solution vectors. The HM is then sorted by their fitness scores as measured by the evaluation function. The N best solution vectors, known as the elite set, are then saved and copied to the HM of the next iteration. NM-SA is then applied to the elite set to generate the $(N+1)$th solution vector of the updated HM. Harmony search is then applied to the whole HM and a ranked selection of this updated HM is then used to replace the remaining $(M-N-1)$ solution vectors of the updated HM. A diagram illustrating one iteration of SHS is shown in Figure 2.8.

SHS was tested using several benchmark optimisation problems with examples of both constrained and unconstrained optimisation. The results were compared with that of the original HS algorithm and with two genetic algorithm variations. All algorithms were allowed to run for 500 function evaluations and the average best results over 50 repetitions were compared. HSA produced slightly more accurate results than all the other algorithms over most of the tested examples. Two real world problems that have been used before to illustrate harmony search namely, a pressure vessel design and a welded beam design, were also optimised using HSA. On both examples HSA produced a better solution than standard harmony search. The authors concluded that HSA performs better than standard HS in both constrained and unconstrained optimisation problems but did not compare results with other modern harmony search variations.

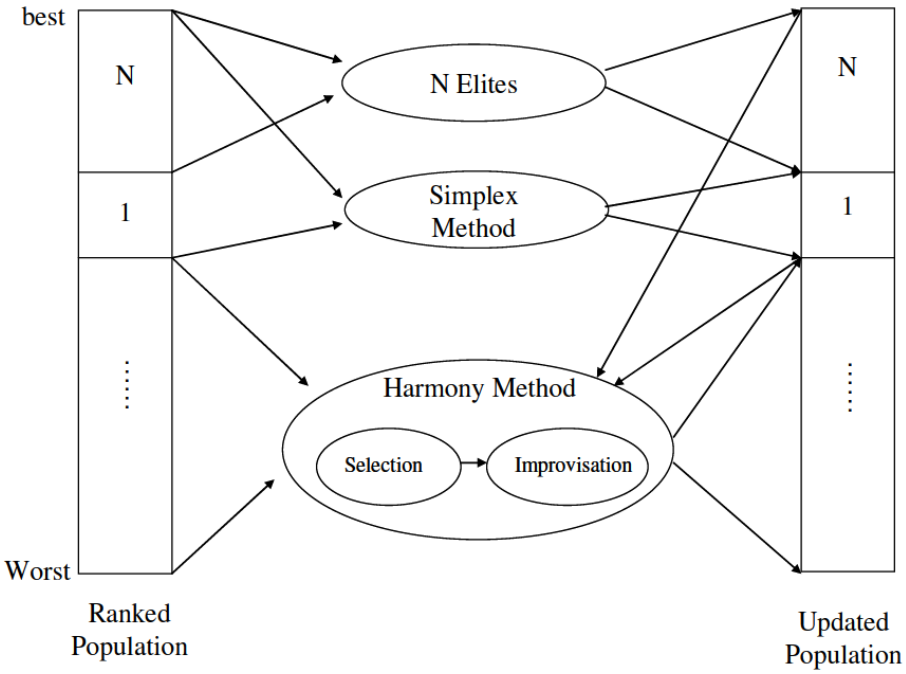

Figure 2.8: This diagram illustrates the way SHS updates the HM using a combination of standard harmony search and NS-SA. The columns on the left and right is the HM before and after the update step (image from [45]).

2.7.2 Harmony Search + Differential Evolution

Li et al. proposed two new hybrid algorithms based on the combination of GHS and the differential evolution algorithm (DE) [46]. The first, called global harmony differential evolution (GHDE), is based on DE but uses the GHS pitch adjustment step to build new solutions. The second, called differential harmony search (DHS), is based on GHS but uses the differential operator from DE to introduce new elements into the HM.

The DE algorithm is a population based optimisation algorithm that has been shown to outperform many other optimisation algorithms in both speed and accuracy [47]. The DE algorithm consists of four main mechanisms namely, parent choice, differential operator, discrete crossover and greedy selection. These operators are very similar to the mutation, crossover and selection operators found in traditional genetic algorithms and work in much the same way. For more details on the DE algorithm see [47].

The GHDE algorithm takes advantage of the GHS pitch adjustment operator to best mimic the highest scoring individual in the population when generating a new

population for use in standard DE. The memory consideration from GHS is also used and together these two operators replace the differential operator step in DE to create the GHDE algorithm.

Instead of focussing on augmenting DE with operators from GHS, the opposite approach is taken with the DHS algorithm. It is based on the GHS algorithm but uses the differential operator from DE to augment the random selection step in GHS. DHS is identical to GHS except for the random selection operator that is randomly replaced by the differential operator. A new parameter called the DR is introduced to control the frequency that random selection is replaced by the differential operator. Additionally, the PAR is kept constant as suggested by the authors of GHS [17].

The performance of both GHDE and DHS was compared with that of DE and GHS using 8 non-linear benchmark functions. All problems were implemented in 30 dimensions and the average error over 100 repetitions were compared for the four algorithms. The maximum number of function evaluations were fixed at 50,000 for all algorithms. It was shown that the hybrid DHS and GHDE algorithms consistently perform better than the GHS and DE algorithms. It was also shown that of the two different approaches to hybridising GHS and DE, DHS is usually the better approach. DHS came closer to the optimal solution in all examples except one in which both DHS and GHDE found the exact optimal solution.

2.7.3 Harmony Search + Sequential Quadratic Programming

Fesanghary et al. [15] noticed that standard harmony search performs well in finding multiple optima in a large and complex search space but is not very efficient in using local search to refine the accuracy of optima in the HM. They proposed to use the sequential quadratic programming (SQP) technique to speed up local search and to improve on the accuracy of harmony search.

Harmony search only considers the objective function during the update of the HM and not during the creation of new solution vectors. By itself, it is therefore considered a blind optimiser since it does not use information about the structure of the objective function like derivative information even when it is available. One strategy to improve on this situation is to add a local optimiser to harmony search that is activated when convergence starts to slow down. The local optimiser uses the optimised solutions in the

HM as starting point and refines them by exploiting the local structure of the objective function around those points.

SQP is the optimisation technique that Fesanghary et al. chose to use as their local optimiser. It was chosen for its computational efficiency and accurate performance over a diverse range of problems [48]. Various constrained and unconstrained benchmark problems were used to test the performance of this hybrid algorithm and results were compared with various other optimisation algorithms including standard harmony search. More real-world examples from structural engineering were also optimised for comparison and the harmony search + SQP hybrid was shown to be superior to most other algorithms in all the examples. However, it is interesting to note that they only report marginally better performance when compared to standard harmony search and IHS. It is also interesting that GHS was mentioned as a improvement to standard harmony search but was not compared with the results from the proposed harmony search + SQP hybrid.

2.7.4 Harmony Search + Particle Swarm Optimisation

In a previous section we introduced the GHS algorithm as an improvement to the original harmony search (see Section 2.6.3). In a way this is already a harmony search + PSO hybrid since the modified pitch adjustment step is identical to the particle update step in PSO. Dr Z.W. Geem proposes a more direct approach called particle swarm harmony search (PSHS) [3].

The PSHS approach is to randomly substitute the memory consideration step with the particle update step from PSO as described in equation (2.38). A new parameter called the particle swarm rate (PSR) controls the rate at which memory consideration is replaced by the new operation.

PSHS was developed for the cost optimisation of water network design and was tested using four benchmark water networks. Unlike the performance comparisons in many of the other algorithms we have considered, the focus of this performance test was on speed as well as accuracy. The performance from a range of algorithms including genetic algorithms, simulated annealing, ant colony optimisation and standard harmony search, were compared with each other. Both the accuracy in finding the optimal solution and the number of function evaluations are compared and it is found that

while both harmony search and PSHS usually find the exact optimum, PSHS does so in considerably less iterations.

It was found that PSHS performs better on the smaller networks where it always reached the optimal solution using the least amount of iterations. As the networks become larger PSHS tends to converge prematurely and its performance becomes similar to that of standard harmony search. This tendency for premature convergence can be controlled using the newly introduced PSR parameter but this is usually done at the cost of speed.

2.7.5 Harmony Search + Simulated Annealing

The simulated annealing (SA) algorithm is a heuristic optimisation algorithm inspired by the annealing process used in metallurgy to increase the size of the crystals in a metal through controlled heating and cooling [32]. It uses an optimisation framework similar to many other heuristic optimisers and is regarded as an adaptation of the Metropolis-Hastings algorithm [49].

A key feature of SA is its ability to not only move towards better solutions by taking advantage of the local structure of the objective function, but also its ability to randomly jump to other possible solutions. This gives it the ability to recover from premature convergence and makes it less likely that it will get stuck in local optima. However, SA suffers from several parameters that are difficult to set and is sensitive to the initial starting point choice that must be set as a possible solution before optimisation can start.

To mitigate some of these disadvantages, Jiang et al. proposes a GHS + SA hybrid algorithm called SAGHS [50]. First the initial population is generated as a HM using random initialisation (see Algorithm 2.2). New solutions are improvised using the GHS scheme but the acceptance of new solutions are decided using the temperature controlled acceptance scheme from SA. The HM is then tested using a sample stability criterion based on the current temperature parameter from SA. If the stability criterion is met the temperature is updated, the annealing process is used to update the best solution and optimisation continues. Once the maximum number of iterations have been reached the best solution is once again updated using the annealing process to produce the final optimal solution.

The performance of SAGHS is tested and compared with SA and GHS using 5 benchmark problems that are commonly used for testing optimisation algorithms. Each algorithm is allowed to run for 50,000 iterations and the final accuracy as well as the number of iterations to get there is compared. In all 5 examples SAGHS achieved the most accurate results using the least amount of iterations and in some cases SA and GHS could not reach a similar accuracy even after 50,000 iterations. The clear conclusion is that a hybrid between SA and GHS can perform better than any of the two do separately.

2.8 Chapter Conclusions

In this chapter we investigated the harmony search algorithm and algorithms based on it. As a metaheuristic algorithm harmony search cannot guarantee that the global optimal solution will be found but often a good estimate is all that is needed. Like other similar metaheuristic algorithms such as SA, PSO and genetic algorithms, harmony search is most useful in situations where traditional optimisation algorithms cannot be used or would take too long to find a good solution.

Many optimisation problems translate to objective functions that are discontinuous, multi-modal, non-differentiable or a combination of these. These are all characteristics that cause traditional gradient based algorithms to fail. Other commonly encountered functions exhibit behaviour that cause gradient based techniques to take impractically long to converge. These are all examples of situations where metaheuristic algorithms are almost exclusively used.

Harmony search is easy to implement, and as we saw can easily be applied and adapted to solve almost any problem that can be modelled as the minimisation or maximisation of an objective function. The objective function itself can be continuous or discrete and a smooth gradient is not required. No initial solutions or carefully chosen starting points are required, in fact the objective function is considered a black box by harmony search. Any procedure that takes as input a solution vector and gives a fitness score as output can be used as an objective function.

These properties make harmony search very attractive and it has been successfully used in a wide range of disciplines including computer vision, vehicle routing, music composition, solving Sudoku and various engineering disciplines[2, 5, 23, 51–53].

While harmony search has proven very effective at avoiding local optima and efficiently covering the search space, the main disadvantage of this algorithms is its weak exploitation of the local search space. This sometimes leads to low accuracy in the final solution. We investigated various approaches to mitigating this shortcoming including the popular GHS algorithm and SAHS which takes an opposite approach (see Sections 2.6.3 and 2.6.4).

In some situations a small HM combined with continuous updates to the HM can lead to a rapid decrease in the diversity of the HM. A loss of diversity in the HM often leads to premature convergence to a local optima. Depending on the size of the optima's basin of attraction, it may take many iterations to escape from this local optima leading to very slow convergence and overall poor performance. One solution to this problem is the DLHS algorithm which we investigated in Section 2.6.5. DLHS attempts to maintain diversity in the HM by dividing it into multiple sub-HM's that converge independently.

Due to its simplicity and ease of implementation, harmony search is easily combined with other algorithms to form harmony search hybrids. This is usually another approach at mitigating some of harmony search's shortcomings, but is also done simply to improve its performance. We investigated five popular harmony search hybrids in Section 2.7 that combine harmony search with proven successful algorithms like SA and PSO. All five report that the hybrid has better performance than both original harmony search and the algorithm that was combined with it. This is by no means an exhaustive list of harmony search hybrids and new ones are being developed continually.

From this I conclude that harmony search is a very flexible and adaptable algorithm. It has been used on its own with large success but can also benefit greatly from the combination with other optimisation approaches. As we saw in Section 2.6, performance can be enhanced by adapting and modifying certain aspects of its design. For example, many derived algorithms like GHS just modify the improvisation step to improve local exploitation. In addition, by dynamically adjusting some parameters during optimisation we not only potentially improve performance, but we also reduce the number of parameters further simplifying implementation. Examples of this include IHS, GHS, SAHS, DLHS and many others that were investigated in this chapter.

In the chapters that follow I introduce four novel algorithms that were all derived from harmony search. Unlike most of the algorithms discussed in this chapter these four

algorithms were adapted to solve a specific problem instead of being designed for generic use. While all four algorithms were developed for use in computer vision applications, we will see that some of the methods developed therein can be used generically.

3

Visual Tracking

> One of the most interesting aspects of the world is that it can be considered to be made up of patterns. A pattern is essentially an arrangement. It is characterised by the order of the elements of which it is made, rather than by the intrinsic nature of these elements .
>
> – *Norbert Wiener*

The aim of visual tracking is to determine the location of some object of interest through a sequence of images. The object of interest is typically modelled as one or more image features combined into a vector called a feature vector. The aim is then to search each frame in the sequence for a specific feature vector and determine its location in the image as image coordinates.

Visual tracking is used in many applications including surveillance [54, 55], driver assistance [56], human-computer interaction [57], augmented reality [58] and video communication [59]. Most of these applications use a video sequence as input and require real-time performance. Exhaustively searching the entire image frame until the image feature is found, is therefore not practical as this approach would be too slow for real-time performance. Instead, an intelligent algorithm is needed that can take advantage of temporal information from previous frames to quickly find the relevant feature vector without having to search the entire frame.

3.1 Problem Statement

Visual tracking methods typically focus on one of two possible approaches called the *top-down* and *bottom-up* approaches. When the focus is on target representation and its localisation we say that it is a bottom-up approach. A bottom-up approach is often used in applications where the target often deforms or changes its appearance. In contrast, a top-down approach focuses on filtering and data association, or in other words, the target dynamics are analysed and used with scene priors to predict the target's location. In applications where real-time performance is important and the target moves around a scene in a way that can be estimated, the focus is usually on a top-down approach.

While the focus may be towards one of these approaches, to some extent both are important and they can therefore also be seen as two components of a tracking algorithm. Focus between the top-down and bottom-up aspects of tracking is dependent on the application, specifically the target's expected movement in the scene and how well it can be modelled as an image feature.

The most popular method for modelling the visual tracking problem is to use the probabilistic state space approach that is used to model discrete-time dynamic systems [60]. The target's location and feature vector representation is represented as a dynamic state sequence $\mathbf{X}_{0:t} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_t\}$ specified by state transition function defined as

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{v}) \ , \tag{3.1}$$

where $\mathbf{v}$ is a noise term that is assumed to be independent and identically distributed (i.i.d). The state vector, $\mathbf{x}$, typically has many components representing the target's location, speed, acceleration and image features. The state transition equation describes a deterministic Markov process. This means that past and future data are statistically independent when we know the current state $\mathbf{x}_t$ [61]. This is known as the *Markov assumption* and implies that the current state, $\mathbf{x}_t$, is only dependent on the previous state, $\mathbf{x}_{t-1}$, instead of the set of past states.

The set of scene measurements, $\mathbf{Z_{0:t}} = \{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_t\}$, are related to the state

sequence by the state measurement equation defined as

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{n}) \ , \tag{3.2}$$

where $\mathbf{n}$ is another noise term that is assumed to be independent and identically distributed (i.i.d).

The aim is then to estimate the current state, $\mathbf{x}_t$, given the all the measurements up to this time, $\mathbf{Z}_{0:t}$. Since this is a statistical process we can equivalently say that we are trying to construct the probability density function (PDF) $p(\mathbf{x}_t \mid \mathbf{Z}_{0:t})$.

One can theoretically solve this problem optimally using the *Bayesian recursive filter* [62]. The filter estimates the posterior PDF, $p(\mathbf{x}_t \mid \mathbf{Z}_{0:t})$, using two steps. In the first step, called the *prediction*, the state transition equation (see equation (3.1)) is used with the posterior PDF of the previous time step, $p(\mathbf{x}_{t-1} \mid \mathbf{Z}_{0:t-1})$, to derive the prior PDF of the current state, $p(\mathbf{x}_t \mid \mathbf{Z}_{0:t-1})$. Next, the *update* step uses the likelihood function, $p(\mathbf{z}_t \mid \mathbf{x}_t)$, of the current measurement together with the prior from the prediction step to derive the posterior PDF for the current time step.

The prediction and update steps are implemented in different ways depending on the assumptions made concerning the linearity of the state transition, f, and state measurement, h, equations as well as the nature of the noise terms, $\mathbf{v}$ and $\mathbf{n}$. For example, if we assume that the noise terms are Gaussian and that f and h are both linear then it has been shown that the optimal solution is implemented by the famous Kalman filter [62]. These highly restrictive assumptions make the Kalman filter impractical in many applications but as we will see in the next section, much has been done to relax some of these restrictions. In the next section an overview is given of the Kalman filter as well as other more modern approaches to the visual tracking problem.

3.2 Overview of Current Approaches

3.2.1 Kalman Filter

The Kalman filter is named after Rudolf E. Kalman who published a recursive solution to the discrete-data linear filtering problem in 1960 [63]. When certain conditions are met the Kalman filter is the optimal estimator of the posterior, $p(\mathbf{x}_t \mid \mathbf{Z}_{0:t})$, in the sense that the *error covariance* is minimised. For this to be true the state transition and

state measurement functions, f and h must be linear and the two noise terms, $\mathbf{v}$ and $\mathbf{n}$, must be zero-mean Gaussian.

When these assumptions are made the state transition and state measurement equations can be defined in matrix notation as follows:

$$\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t + \mathbf{v} \tag{3.3}$$

$$\mathbf{z}_t = \mathbf{H}\mathbf{x}_t + \mathbf{n} \tag{3.4}$$

where $\mathbf{F}$ is the state transition matrix and $\mathbf{H}$ is the state measurement matrix. Since $\mathbf{v}$ and $\mathbf{n}$ are zero-mean Gaussian random variables we define their PDFs as

$$p(\mathbf{v}) \backsim \mathcal{N}(0, \mathbf{Q}) \tag{3.5}$$

$$p(\mathbf{n}) \backsim \mathcal{N}(0, \mathbf{R}) \tag{3.6}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are the process and noise covariance matrices respectively.

With these definitions one can illustrate the Kalman filter using a hidden Markov model (HMM). HHMs are frequently used to model sequential data statistically with each node in the sequence representing a state at a certain time step. In this section the states are represented by the state vector $\mathbf{x}_t$ that is modelled as a Gaussian PDF with the state mean as $\overline{\mathbf{x}}_t$ and the state covariance matrix as $\mathbf{P}$. Since Gaussian distributions are assumed, each state is fully described by the mean and covariance matrix. An illustration of the Kalman filter as a HMM is shown in Figure 3.1.

As mentioned in the previous section, the Kalman filter is a special implementation of the Bayesian recursive filter and therefore also consists of a predict and update phase. For the purpose of clarity we now adopt the simplified notation introduced by Jordan [64]. Let the mean of $\mathbf{x}_{t+1}$ conditioned on $\mathbf{Z}_{0:t}$ be written as $\overline{\mathbf{x}}_{t+1|t}$ and in the same way the state covariance matrix is written as $\mathbf{P}_{t+1|t}$[1]. With this notation the predict step needs to calculate values for $\overline{\mathbf{x}}_{t+1|t}$ and $\mathbf{P}_{t+1|t}$ which is done using the following equations (for the full derivation see [63]).

$$\overline{\mathbf{x}}_{t+1|t} = \mathbf{F}\mathbf{x}_{t|t} \tag{3.7}$$

$$\mathbf{P}_{t+1|t} = \mathbf{F}\mathbf{P}_{t|t}\mathbf{F}^T + \mathbf{Q} \tag{3.8}$$

[1] Jordan uses the $\hat{}$ symbol to indicate the mean but I use the $\bar{}$ to maintain a consistent notation throughout the thesis.

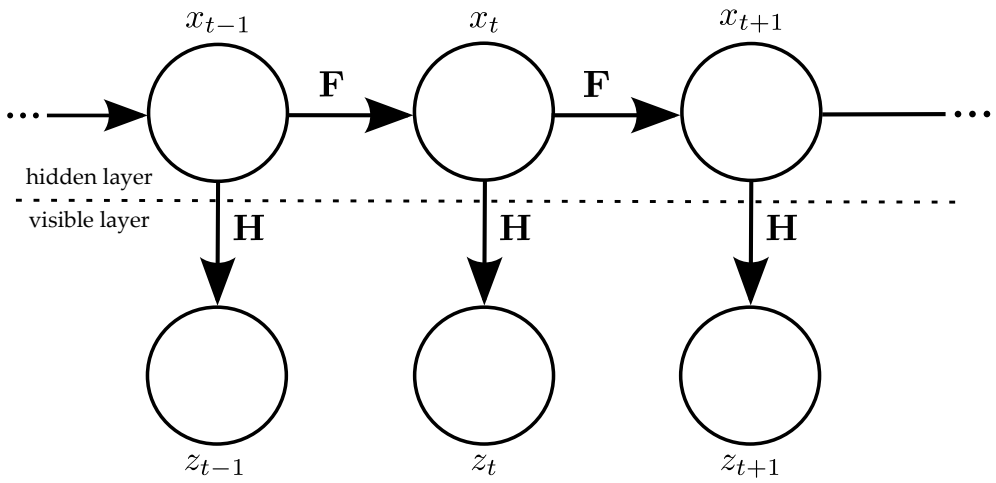

Figure 3.1: This is the Kalman filter illustrated as a HMM. The hidden nodes are the states, x_t while the z_t nodes represent the observable data called the measurements. State transitions are controlled by the matrix $\mathbf{F}$ while the relationship between the state and the observations are controlled by $\mathbf{H}$.

In the update step the posterior is calculated based on the prior from the predict step and the latest observation.

$$\overline{\mathbf{x}}_{t+1|t+1} = \overline{\mathbf{x}}_{t+1|t} + \mathbf{K}(\mathbf{z}_{t+1} - \mathbf{H}\overline{\mathbf{x}}_{t+1|t}) \tag{3.9}$$

$$\mathbf{P}_{t+1|t+1} = \mathbf{P}_{t+1|t} - \mathbf{K}\mathbf{H}\mathbf{P}_{t+1|t} \ , \tag{3.10}$$

where $\mathbf{K}$ is called the *Kalman gain matrix* determined by

$$\mathbf{K}_{t+1} = \mathbf{P}_{t+1|t}\mathbf{H}^T(\mathbf{H}\mathbf{P}_{t+1|t}\mathbf{H}^T + \mathbf{R})^{-1} \tag{3.11}$$

Together, the predict and update steps form a recursive loop of Figure 3.2 that is easy to implement and computationally efficient. However, before the Kalman filter can be implemented one needs detailed knowledge of the target's dynamics and the measurement system. The $\mathbf{F}$ and $\mathbf{H}$ matrices model the system dynamics and need to be accurately determined to ensure accurate estimates of the system state. The system and measurement noise covariances, $\mathbf{Q}$ and $\mathbf{R}$, must also be known as well as the initial state covariance, $\mathbf{P}_0$. The matrices must be accurately determined or estimated before filtering can start.

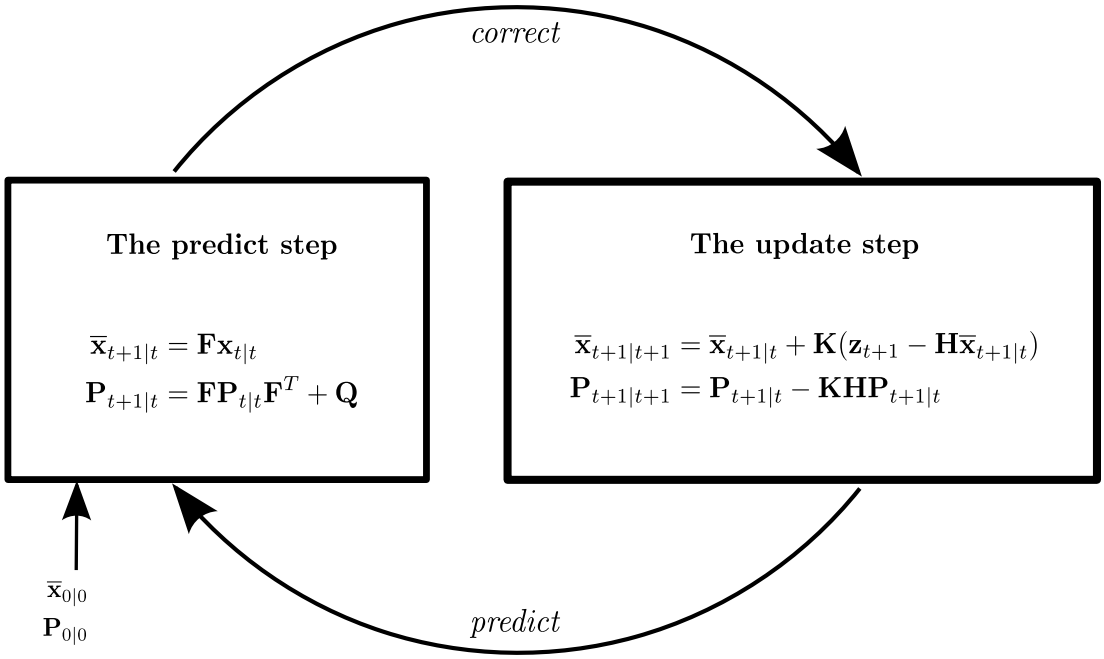

Figure 3.2: This diagram shows the main recursive loop of the Kalman filter. First the prior is predicted in the *predict* step. Then the predicted prior is corrected using the latest observation in the *update* step.

3.2.2 Extended Kalman Filter

In systems that can be modelled using linear equations the Kalman filter is the best estimator possible but in most real applications accurate modelling is only possible with non-linear equations. The extended Kalman filter (EKF) aims to address the limitation by linearising the system dynamics using a Taylor expansion [65]. The state transition and measurement equations are now defined as general differentiable functions with added zero-mean Gaussian noise.

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t) + \mathbf{v} \tag{3.12}$$

$$\mathbf{z}_{t+1} = h(\mathbf{x}_{t+1}) + \mathbf{n} \tag{3.13}$$

Now f is linearised using a first order order Taylor expansion around $\overline{\mathbf{x}}_{t|t}$ to give

$$f(\mathbf{x}_t) = f(\overline{\mathbf{x}}_{t|t}) + \left.\frac{\partial f(\mathbf{x}_t)}{\partial \mathbf{x}_t}\right|_{\mathbf{x}_t = \overline{\mathbf{x}}_{t|t}} (\mathbf{x}_t - \overline{\mathbf{x}}_{t|t}) + \text{HOT} + \mathbf{v}\ , \tag{3.14}$$

where HOT are the higher order terms that are ignored in the first order expansion. Since $\mathbf{x}_t$ is a vector the partial derivatives form the Jacobian matrix indicated with the ∇ symbol. The EKF uses the following Jacobian matrices to derive values for the mean and covariances.

$$\mathbf{F}_{t|t} = \nabla f(\mathbf{x}_t)|_{\overline{\mathbf{x}}_{t|t}} \tag{3.15}$$

$$\mathbf{H}_{t+1|t} = \nabla h(\mathbf{x}_t)|_{\overline{\mathbf{x}}_{t+1|t}} \tag{3.16}$$

With these definitions the predict step becomes

$$\overline{\mathbf{x}}_{t+1|t} = f(\overline{\mathbf{x}}_{t|t}) \tag{3.17}$$

$$\mathbf{P}_{t+1|t} = \mathbf{F}_{t|t}\mathbf{P}_{t|t}\mathbf{F}_{t|t}^T + \mathbf{Q}\ , \tag{3.18}$$

with $\mathbf{Q}$ again defined as the system noise covariance. This result is very similar to the linear Kalman predict step and the same can be said for the update step that is as follows:

$$\overline{\mathbf{x}}_{t+1|t+1} = \overline{\mathbf{x}}_{t+1|t} + \mathbf{K}[\mathbf{z}_{t+1} - h(\overline{\mathbf{x}}_{t+1|t})] \tag{3.19}$$

$$\mathbf{P}_{t+1|t+1} = \mathbf{P}_{t+1|t} - \mathbf{K}\mathbf{H}_{t+1|t}\mathbf{P}_{t+1|t}\ , \tag{3.20}$$

where $\mathbf{K}$ is called the *Kalman gain matrix* determined by

$$\mathbf{K}_{t+1} = \mathbf{P}_{t+1|t}\mathbf{H}_{t+1|t}^T(\mathbf{H}_{t+1|t}\mathbf{P}_{t+1|t}\mathbf{H}_{t+1|t}^T + \mathbf{R})^{-1} \tag{3.21}$$

For a full derivation of these equations see [66]. It is important to notice that the EKF is not optimal like the linear Kalman filter. It only uses a linearised approximation of f and h. The EKF suffers from poor performance when f and h are highly nonlinear and can diverge after several predict-update cycles when the first order Taylor expansion of f and h is not a good approximation. Moreover, the $\mathbf{F}$ and $\mathbf{H}$ matrices now depend on previous state estimates and must be calculated again during each predict-update step. However, there are more accurate ways of approximating the system dynamics as we will see in the next section.

3.2.3 Unscented Kalman Filter

Unlike the EKF the unscented Kalman filter (UKF) does not linearise the state transition and measurement functions but uses the true nonlinear functions and instead approximates the state variable's distribution [67]. When the effects of the higher order terms in the Taylor series expansion becomes significant the local linearity assumption that the EKF uses breaks down resulting in large errors that grow in time. The UKF idea is that it is easier to approximate a Gaussian random variable than it is to approximate a general nonlinear function.

The state variable's distribution, $p(\mathbf{x})$, is approximated using a set of deterministically chosen weighed sample points called *sigma* points. The sigma points are chosen so they both completely capture the mean and covariance of the prior's distribution, as well as the posterior's mean and convergence to the 2nd order when that the prior's sigma points are propagated through the true nonlinear state transition function [68]. Errors are only introduced in the 3rd and higher orders but can be scaled away to a large extent by fine tuning a scaling parameter.

The sigma points and their weights are chosen using the unscented transform. The UKF requires $2n+1$ sigma points to fully capture the distribution's statistics where n is the number of components in $\mathbf{x}$. The unscented transform is a method for calculating the mean and covariance of a random variable that undergoes a nonlinear transformation. Consider the random variable $\mathbf{x}$ propagated through the nonlinear function $g(\mathbf{x})$ to create the random variable $\mathbf{y}$. If we assume that $\overline{\mathbf{x}}$ is the mean of $\mathbf{x}$ and $\mathbf{P}_x$ is the covariance, the weighted sigma points for $\mathbf{x}$ are calculated as follows:

$$\begin{aligned}
\mathcal{X}_0 &= \overline{\mathbf{x}} & w_0 &= \frac{\kappa}{n+\kappa} & i &= 0 \\
\mathcal{X}_i &= \overline{\mathbf{x}} + \left(\sqrt{(n+\kappa)\mathbf{P}}\right)_i & w_i &= \frac{1}{2(n+\kappa)} & i &= 1,\ldots,n \\
\mathcal{X}_i &= \overline{\mathbf{x}} - \left(\sqrt{(n+\kappa)\mathbf{P}}\right)_i & w_i &= \frac{1}{2(n+\kappa)} & i &= n+1,\ldots,2n
\end{aligned} \tag{3.22}$$

where κ is the scaling parameter and $\left(\sqrt{(n+\kappa)\mathbf{P}}\right)_i$ is the ith row of the matrix square root of $(n+\kappa)\mathbf{P}$. If we now propagate each of the sigma points through the nonlinear function $g(\mathbf{x})$ we get

$$\mathcal{Y}_i = g(\mathcal{X}_i) \qquad i = 0,\ldots,2n \tag{3.23}$$

which allows us to calculate a mean and covariance for $\mathcal{Y}$ as

$$\begin{aligned}\overline{\mathbf{y}} &= \sum_{i=0}^{2n} w_i \mathcal{Y}_i \\ \mathbf{P}_y &= \sum_{i=0}^{2n} w_i (\mathcal{Y}_i - \overline{\mathbf{y}})(\mathcal{Y}_i - \overline{\mathbf{y}})^T \ .\end{aligned} \tag{3.24}$$

For the full derivation of the sigma points see [67].

The UKF is then implemented by applying the unscented transform to the standard recursive Bayesian filter mentioned in Section 3.1. In order to incorporate the system and measurement noise a new augmented state is created by concatenating the original state and the noise variables. We refer to this new augment state as $\mathbf{x}_t^a = [\mathbf{x}_t^T \ \mathbf{v}^T \ \mathbf{n}^T]^T$ with the initial mean defined as $\overline{\mathbf{x}}_0^a = [\overline{\mathbf{x}}_0^T \ \mathbf{0}^T \ \mathbf{0}^T]^T$. The initial state covariance is then similarly defined as

$$\mathbf{P}_0^a = E[(\mathbf{x}_t^a - \overline{\mathbf{x}}_t^a)(\mathbf{x}_t^a - \overline{\mathbf{x}}_t^a)^T] = \begin{bmatrix} \mathbf{P}_0 & 0 & 0 \\ 0 & \mathbf{Q} & 0 \\ 0 & 0 & \mathbf{R} \end{bmatrix} \ . \tag{3.25}$$

Sigma points are then calculated from the augmented state and propagated through the state transition and measurement equations to update the posterior. The full UKF algorithm for the calculation of the posterior is as follows:

1. Calculate sigma points for predict step:

$$\boldsymbol{\mathcal{X}}_t^a = \left[\overline{\mathbf{x}}_t^a \ \ \overline{\mathbf{x}}_t^a \pm \sqrt{(n+\kappa)\mathbf{P}_t^a} \right] \tag{3.26}$$

2. Predict step:

$$\boldsymbol{\mathcal{X}}^x_{t+1|t} = f(\boldsymbol{\mathcal{X}}^x_t, \boldsymbol{\mathcal{X}}^v_t) \tag{3.27}$$

$$\overline{\mathbf{x}}_{t+1|t} = \sum_{i=0}^{2n} w_i \mathcal{X}^x_{i,t+1|t} \tag{3.28}$$

$$\mathbf{P}_{t+1|t} = \sum_{i=0}^{2n} w_i [\mathcal{X}^x_{i,t+1|t} - \overline{\mathbf{x}}_{t+1|t}][\mathcal{X}^x_{i,t+1|t} - \overline{\mathbf{x}}_{t+1|t}]^T \tag{3.29}$$

$$\boldsymbol{\mathcal{Y}}_{t+1|t} = h(\boldsymbol{\mathcal{X}}^x_{t+1|t}, \boldsymbol{\mathcal{X}}^n_t) \tag{3.30}$$

$$\overline{\mathbf{y}}_{t+1|t} = \sum_{i=0}^{2n} w_i \mathcal{Y}_{i,t+1|t} \tag{3.31}$$

3. Update step:

$$\mathbf{P}_{\overline{\mathbf{y}}_{t+1}\overline{\mathbf{y}}_{t+1}} = \sum_{i=0}^{2n} w_i [\mathcal{Y}_{i,t+1|t} - \overline{\mathbf{y}}_{t+1|t}][\mathcal{Y}_{i,t+1|t} - \overline{\mathbf{y}}_{t+1|t}]^T \tag{3.32}$$

$$\mathbf{P}_{\overline{\mathbf{x}}_{t+1}\overline{\mathbf{y}}_{t+1}} = \sum_{i=0}^{2n} w_i [\mathcal{X}^x_{i,t+1|t} - \overline{\mathbf{x}}_{t+1|t}][\mathcal{Y}_{i,t+1|t} - \overline{\mathbf{y}}_{t+1|t}]^T \tag{3.33}$$

$$\mathbf{K}_{t+1} = \mathbf{P}_{\overline{\mathbf{x}}_{t+1}\overline{\mathbf{y}}_{t+1}} \mathbf{P}^{-1}_{\overline{\mathbf{y}}_{t+1}\overline{\mathbf{y}}_{t+1}} \tag{3.34}$$

$$\overline{\mathbf{x}}_{t+1|t+1} = \overline{\mathbf{x}}_{t+1|t} + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \overline{\mathbf{y}}_{t+1|t}) \tag{3.35}$$

$$\mathbf{P}_{t+1|t+1} = \mathbf{P}_{t+1|t} - \mathbf{K}_{t+1} \mathbf{P}_{\overline{\mathbf{y}}_{t+1}\overline{\mathbf{y}}_{t+1}} \mathbf{K}^T_{t+1} \ , \tag{3.36}$$

where $\boldsymbol{\mathcal{X}}^a_t = [(\boldsymbol{\mathcal{X}}^x_t)^T \ \ (\boldsymbol{\mathcal{X}}^v_t)^T \ \ (\boldsymbol{\mathcal{X}}^n_t)^T]^T$.

The UKF has several advantages over the EKF. Not only is it much more accurate but it does not require the calculation of Jacobian matrices. However, it does require the calculation of a matrix square root. This can be done very efficiently using Cholesky factorisation. Since we are working with covariance matrices that can be updated recursively, the Cholesky factorisation can be done recursively as well making it an order n^2 operation (for details on how this is done see [67]). This means that the UKF can be performed at the same computational cost as the EKF even though it performs much better.

There is even a way to improve the performance of the UKF to be better than the EKF in the special case when the system and measurement noise is purely additive (this is often the case). In this case the system state does not need to be augmented with the Q and R covariances reducing the number of dimensions in the filter. This also means that fewer sigma points need to be calculated. Further details of this modification can be found in [67]. The UKF has been used successfully in visual tracking and proved to be superior to the EKF in practical experiments [69].

3.2.4 Particle Filter

While the EKF and the UKF both allow the Kalman filter to be used in non-linear systems, the assumption is still made that the states can be modelled as Gaussian. This assumption cannot always be safely made, especially when there are multiple sensors involved. The particle filter is a *sequential Monte Carlo* method that does not make this assumption [70, 71].

In recent years the particle filter has been extensively used in tracking and robotics applications [54, 72–74]. The three main reasons for this popularity is as follows:

- **Computational efficiency** Some algorithms make it possible to calculate the belief in polynomial time complexity but most algorithms operate in the region of quadratic to cubic time complexity [61]. Particle filter based techniques offer the flexibility of being an *any-time* algorithm, enabling them to trade off accuracy with computational efficiency depending on the specific application and hardware environment.

- **Accuracy** Many algorithms only approximate a certain class of distributions accurately. For example, the Kalman filter family assumes that the system state can be approximated as a Gaussian distribution. This approximation is often inaccurate as the system state is often multi-modal and cannot accurately be approximated as a unimodal Gaussian distribution [75]. Particle filter representations can approximate a wide variety of distributions, however, the number of particles needed to accurately approximate certain distributions may be too large to be practical [61].

- **Ease of Implementation** One of the main reasons for the popularity of particle filters is the ease with which one can implement a particle filter that is capable of accurately approximating complex non-linear distributions. No complex mathematical operations like the calculation of Jacobian matrices or the matrix square root are required.

As mentioned, the particle filter is a sequential Monte Carlo method. As such it uses a statistical sampling method to represent the system state. Instead of representing the state distribution as a mean and variance from a Gaussian variable, as is the case with the Kalman filter, the distribution is represented with a number of weighted samples known as particles. However, in practice it is seldom possible to sample the state distribution directly and the classical solution is using the *importance sampling* method and sampling from an alternate distribution called the *proposal* distribution.

If one could sample from the posterior state distribution directly one could estimate the mean of the posterior with N particles using perfect Monte Carlo simulation [68, 70].

$$\overline{p}(\mathbf{x}_{0:t}|\mathbf{z}_{0:t}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{x}_{0:t} - \mathbf{x}_{0:t}^{(i)}) \tag{3.37}$$

where the random samples $\{\mathbf{x}_{0:t}^{(1)}, \mathbf{x}_{0:t}^{(2)}, \ldots, \mathbf{x}_{0:t}^{(N)}\}$ are independent and identically distributed (i.d.d) from the posterior distribution $p(\mathbf{x}_{0:t}|\mathbf{z}_{0:t})$ and δ denotes the Dirac delta function. Consequently, we can estimate the expectation of the distribution after propagation through some non-linear function $g(\mathbf{x}_{0:t})$ using

$$E[g(\mathbf{x}_{0:t})] \approx \frac{1}{N}\sum_{i=1}^{N} g(\mathbf{x}_{0:t}^{(i)}) \tag{3.38}$$

Since we cannot sample from the posterior directly, we circumvent the difficulty by sampling from the easy-to-sample proposal distribution, denoted by $q(\mathbf{x}_{0:t}|\mathbf{z}_{0:t})$, and make the following substitution:

$$E(g(\mathbf{x}_{0:t})) = \int g(\mathbf{x}_{0:t})\frac{p(\mathbf{x}_{0:t}|\mathbf{z}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{z}_{0:t})}q(\mathbf{x}_{0:t}|\mathbf{z}_{0:t})d\mathbf{x}_{0:t} \tag{3.39}$$

$$= \int g(\mathbf{x}_{0:t})\frac{p(\mathbf{x}_{0:t}|\mathbf{z}_{0:t})p(\mathbf{x}_{0:t})}{p(\mathbf{z}_{0:t})q(\mathbf{x}_{0:t}|\mathbf{z}_{0:t})}q(\mathbf{x}_{0:t}|\mathbf{z}_{0:t})d\mathbf{x}_{0:t} \tag{3.40}$$

$$= \int g(\mathbf{x}_{0:t})\frac{w_t(\mathbf{x}_{0:t})}{p(\mathbf{z}_{0:t})}q(\mathbf{x}_{0:t}|\mathbf{z}_{0:t})d\mathbf{x}_{0:t}\ , \tag{3.41}$$

where $w_t(\mathbf{x}_{0:t})$ are the importance weights calculated using the following equation.

$$w_t(\mathbf{x}_{0:t}) = \frac{p(\mathbf{z}_{0:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{z}_{0:t})} \ . \tag{3.42}$$

The expectation can then be approximated by the following estimate:

$$\overline{E(g(\mathbf{x}_{0:t}))} = \sum_{i=1}^{N} g(\mathbf{x}_{0:t}^{(i)})\widetilde{w}_t(\mathbf{x}_{0:t}^{(i)}) \ , \tag{3.43}$$

where the normalized importance weights $\widetilde{w}_t^{(i)}$ are given by

$$\widetilde{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^{N} w_t^{(j)}} \ . \tag{3.44}$$

According to the law of large numbers, we then have that

$$\overline{E(g_t(\mathbf{x}_{0:t}))} \xrightarrow[N\to\infty]{a.s.} E(g_t(\mathbf{x}_{0:t})) \ , \tag{3.45}$$

where $\xrightarrow[N\to\infty]{a.s.}$ denotes almost sure convergence. For the full derivation of this result see [68].

Since we wish to approximate the state distribution in a sequential recursive manner without modifying past states, we choose the proposal density to fit the following form:

$$q(\mathbf{x}_{0:t}|\mathbf{x}_{0:t}) = q(\mathbf{x}_{0:t-1}|\mathbf{x}_{0:t-1})q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{0:t}) \ . \tag{3.46}$$

Here we assume that the states correspond to a Markov process and that observations are conditionally independent given the states. Then we get that

$$p(\mathbf{x}_{0:t}) = p(\mathbf{x}_0)\prod_{j=1}^{t} p(\mathbf{x}_j|\mathbf{x}_{j-1}) \qquad \text{and} \qquad p(\mathbf{z}_{0:t}|\mathbf{x}_{0:t}) = \prod_{j=1}^{t} p(\mathbf{z}_j|\mathbf{x}_j) \ . \tag{3.47}$$

A recursive estimate for the importance weights can then be derived (see [68]) and is given by

$$w_t = w_{t-1}\frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t})} \tag{3.48}$$

With this we have a sequential mechanism to update importance weights given

an appropriate proposal distribution, $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t})$. This procedure is known as *sequential importance sampling* and it allows us to recursively calculate estimates of the form seen in equation (3.43) [62, 68, 70].

The exact form of the proposal density is a critical design issue. The perfect form is usually difficult to implement or computationally expensive to sample from and is thus usually approximated for the sake of easy sampling. It has been shown that the best proposal distributions are those that minimise the variance of the importance weights [70]. Doucet et al. proved that the following proposal distribution is optimal and minimises the variance of the importance weights [76].

$$q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{0:t}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_{0:t}) \tag{3.49}$$

The choice of proposal density is a distinguishing feature for many particle filter implementations and many researchers have commented on the optimization of this choice [20, 77–79]. The most popular (but by no means optimal) choice of proposal is the transition prior

$$q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{0:t}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{3.50}$$

Although this choice results in higher variance in the importance weights it is often the easiest proposal to implement [68]. For example, when the system noise is assumed to be Gaussian and additive the transition prior is simply a Gaussian distribution around the state transition function.

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(f(\mathbf{x}_{t-1}, 0), \mathbf{Q}) \tag{3.51}$$

The calculation of the importance weights is also simplified when this proposal distribution is used. Only the likelihood, $p(\mathbf{z}_t|\mathbf{x}_t)$, which is determined by the measurement equation must now be calculated.

$$w_t = w_{t-1}\frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t})} \tag{3.52}$$

$$= w_{t-1}\frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{p(\mathbf{x}_t|\mathbf{x}_{t-1})} \tag{3.53}$$

$$= w_{t-1}p(\mathbf{z}_t|\mathbf{x}_t) \tag{3.54}$$

However, the main problem with this choice is that it fails to incorporate the latest available observation data. The result is that only a few particles will have significant importance weights when they are evaluated using the likelihood. If the likelihood happens to lie in one of the tails of the prior distribution or if it is too narrow (low measurement error), this problem is compounded and normally leads to serious degeneracy of particles and ultimately a poor approximation of the system state. It is therefore important that the proposal distribution is chosen so that particles are moved towards regions of high likelihood. Figure 3.3 illustrates this situation.

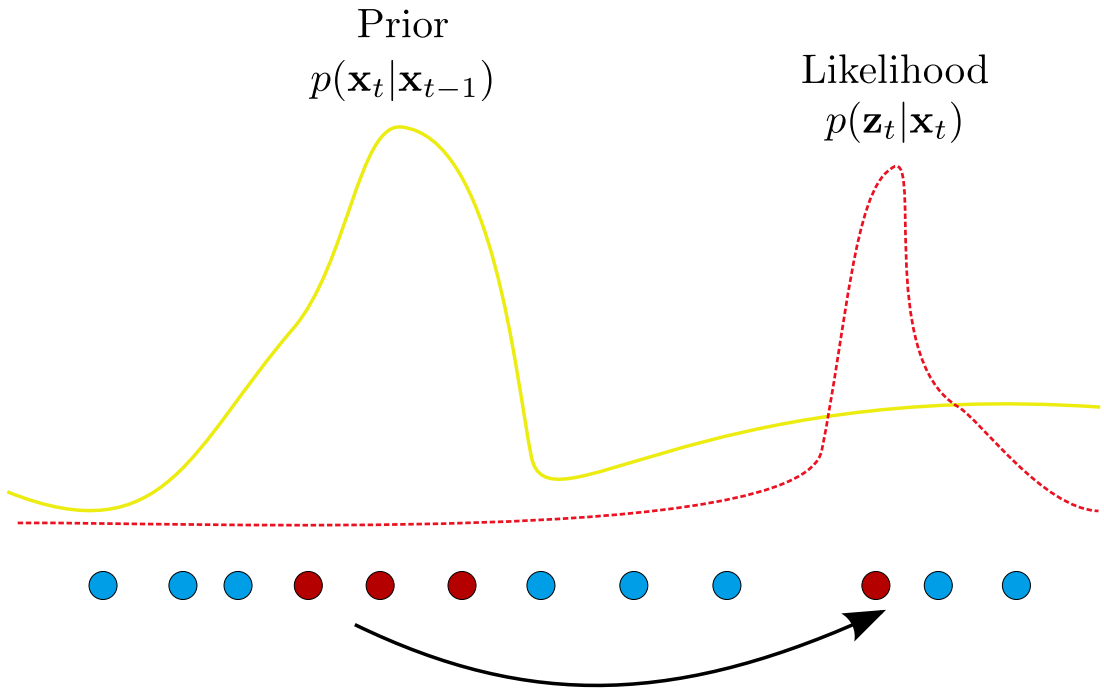

Figure 3.3: This diagram illustrates what happens when particles (represented by the blue and red dots) are sampled from the prior distribution. If the likelihood is too narrow compared to the prior distribution only a few particles will end up with a significant importance weight. The red dots indicate the particle that will have a high fitness assigned to them.

The most common strategy used to solve the degeneracy problem is to include an extra selection step [70]. Several selection schemes have been proposed in particle filter literature and the idea is to associate a number of *children*, say $N_i \in \mathbb{N}$ to each particle such that $\sum_{i=1}^{N} N_i = N$.

The schemes are all designed to satisfy $E(N_i) = N\widetilde{w}_t^{(i)}$ but their performance varies in terms of the variance of the particles. One of the most common schemes is the sampling-importance resampling (SIR) algorithm introduced by Gordon et al. [80]. It involves mapping the weighted set of particles $\{\mathbf{x}_{0:t}^{(i)}, \widetilde{w}_t^{(i)}\}$ into an equally weighted set $\{\mathbf{x}_{0:t}^{(j)}, N^{-1}\}$ with some of the particles cloned and others rejected from the set. This can be done by uniformly sampling from the discrete set $\{\mathbf{x}_{0:t}^{(i)}, i = 1, \ldots, N\}$ with probabilities $\{\widetilde{w}_t^{(i)}, i = 1, \ldots, N\}$. For more details on how this can be effectively accomplished see [68]. SIR is easy to implement and works well enough for most applications but other sampling strategies like *residual sampling* and *minimum variance sampling* report better results without adding computational complexity. For further details on these and other sampling strategies see [62].

However, resampling eventually causes the particle weights to collapse into a singe point degrading the ability of the particle filter to accurately model the distribution. A brute force method of solving this problem is simply to increase the number of particles but there are better methods [68], [70] and [61]. Notice also that resampling at every iteration is not always necessary and only resampling at certain intervals delays the problem without affecting the accuracy of the filter.

A diagram illustrating one full iteration of the particle filter is shown in Figure 3.4. First, the proposal distribution is sampled to generate a particle set that approximates of the prior distribution. The particles are then assigned importance weights to create a weighted particle set. Then the particles are resampled based on their weights to create an unweighted set with some particles cloned and others rejected. This set is then diffused to introduce particle variety using some selection scheme. After weighing the resulting set using the latest observation we get the desired approximation of the posterior.

Thus far we have only discussed the simplest of particle filter designs. Many aspects of the algorithm can be improved, for example, the proposal distribution and the resampling step. By combining the Gaussian approximations of the EKF with the particle filter one can create a particle filter that uses the EKF to create much better proposal distributions that take the latest observation into account. The resulting algorithm is called the *extended Kalman particle filter* [68].

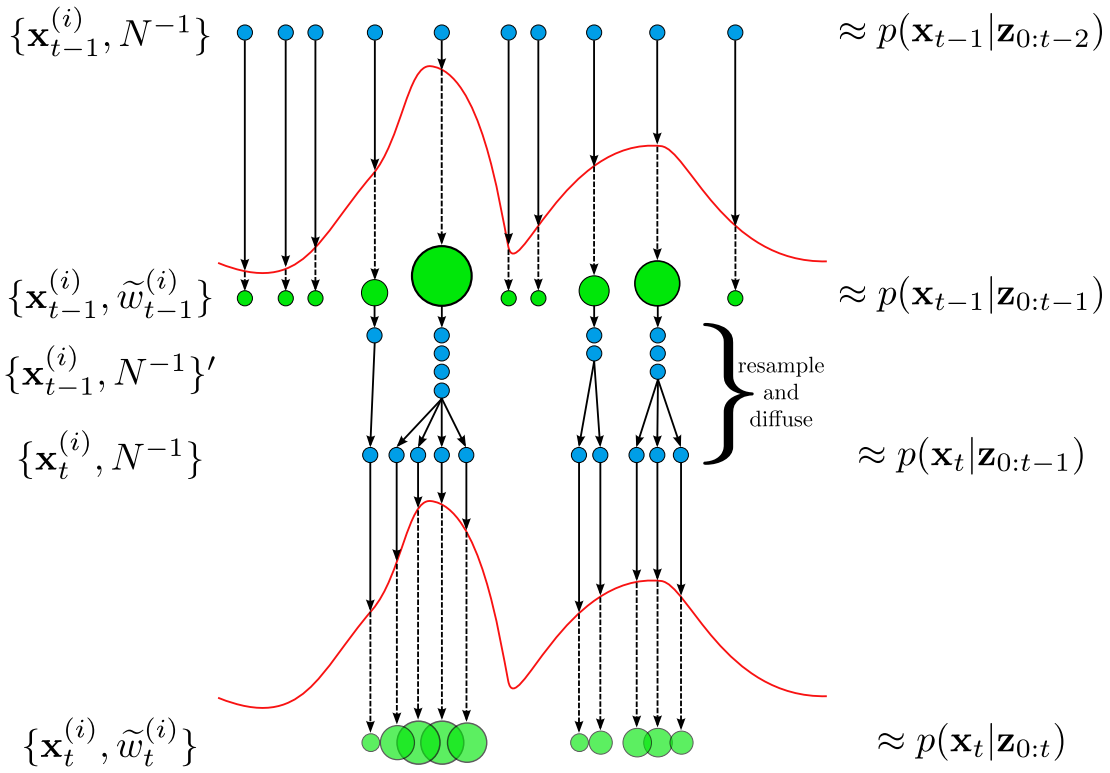

Figure 3.4: In this diagram a particle filter starts with the unweighed particle set $\{\mathbf{x}_{t-1}^{(i)}, N^{-1}\}$ which is an approximation of $p(\mathbf{x}_{t-1}|\mathbf{z}_{0:t-2})$. The dots represent particles with the blue dots as unweighted particles and the green ones weighted. Importance weights are then calculated for each particle using the measurements from time $t-1$ (the largest dots represent the highest fitness). The resulting weighted measure, $\{\mathbf{x}_{t-1}^{(i)}, \widetilde{w}_{t-1}^{(i)}\}$, is an approximation of $p(\mathbf{x}_{t-1}|\mathbf{z}_{0:t-1})$. The resample step then clones the best particles and rejects the worst ones creating an unweighted particle set, $\{\mathbf{x}_{t-1}^{(i)}, N^{-1}\}'$. The particles are then diffused using some selection scheme to introduce diversity resulting in the unweighted set $\{\mathbf{x}_{t}^{(i)}, N^{-1}\}$, which is an approximation of $p(\mathbf{x}_{t}|\mathbf{z}_{0:t-1})$. Finally an approximation for the posterior, $p(\mathbf{x}_{t}|\mathbf{z}_{0:t})$, is created when this set is weighed using the latest measurement, resulting in the particle set $\{\mathbf{x}_{t}^{(i)}, \widetilde{w}_{t}^{(i)}\}$.

However, one can take this one step further and improve even more. We already know from the discussion of the UKF (see Section 3.2.3) that the UKF gives a much better approximation of a distribution that undergoes a nonlinear transformation. One can therefore use the UKF to generate a proposal distribution that is much closer to the true state distribution. Each particle is propagated using the UKF resulting in a weighted set of particles that effectively uses the most recent observation and gives an accurate approximation of the posterior. When the UKF and the particle filter are combined in this way the resulting algorithm is called the *unscented particle filter* (UPF) [68]. Since the particles are used more effectively the filter does not require many particles and the UPF may even be more computationally efficient than the basic particle filter.

3.3 Tracking as an Optimisation Problem

Thus far we have only looked at the visual tracking problem as a probabilistic state space that is approximated using different implementations of the Bayesian recursive filter. Visual tracking can also be modelled as a more general optimisation problem.

Consider that visual tracking is essentially the successive localisation of a specific region of interest in an image through a video sequence. The use of temporal information by considering the input as a *sequence* of correlated images is what differentiates tracking from simply using successive localisation. This means that images are not treated in isolation from their neighbours in the sequence. Instead, the motion of the target (region of interest) between images in the sequence is modelled which aides in the prediction of the target's position in subsequent images.

The region of interest captures the target we are tracking and distinguishes it from the background. To accomplish this the target has to be accurately modelled in a way that allows us to measure how close the region of interest is in completely capturing the target without capturing any of the background. This measurement can then be considered as an objective function that when optimised provides the definition of the region of interest that best captures the target.

Based on this approach one can construct a complete visual tracking system with the following three components:

- **The target model** The target model has to accurately capture the features that are unique to the target so it can be robustly distinguished from other objects in the image and the background. This is a challenging criterion as dramatic changes in the target's appearance is a common occurrence in real video sequences. For example, if the target is a person running through a scene, the running motion will cause severe geometric deformation of the target causing changes in shape that are difficult to predict. Partial occlusion is another common occurrence that can severely affect the target's appearance. Returning to the example of the running person, if a cyclist should pass in front of the man, partially occluding him from the view of the camera, a non-robust target model will likely fail to recognise the man as the original target. These and other factors like dynamic changes in illumination, reflective surfaces and non-target objects that are similar in appearance to the target, make the choice of target model critical to the accuracy of the tracking system.

- **The objective function** The objective function is a metric that measures how close the localised region fits the target model. Since this function will be evaluated many times by the optimisation algorithm before an optimum is found, it is very important that it be as computationally efficient as possible. However, it is even more important that it accurately measures the fitness of a candidate localisation. Ideally, one would like it to be unimodal with with a large basin of attraction surrounding the optimum point. In other words, the true best solution (in this case the perfect localisation) should be the global optimum in an objective function without any local optima. While this function would be very easy to optimise, it is very rarely possible to construct such a function in a real tracking situation. Instead, one usually estimates the ideal and rely on the robustness of the optimisation algorithm to avoid local optima and adjust the localisation when the global optima no longer represents the perfect localisation.

- **The optimisation algorithm** The optimisation algorithm is responsible for finding the optimum solution in the objective function. In Chapter 2 I briefly mentioned several optimisation algorithms including a detailed discussion of the harmony search algorithm. These are generic algorithms that can be used to find the fittest solution given a search space and an objective function. However, due

to imperfect modelling of the target and imperfect approximations of the objective function I rely on the optimisation algorithm to rely on auxiliary information like the target's perceived speed and acceleration to compensate. The optimisation function should ideally guarantee convergence to the perfect solution and be bounded in computation time. However, like the objective function, this ideal situation is rarely possible. One usually relies on an acceptable compromise between accuracy and speed that gives the best estimate of the target's location given the available computational resources.

3.3.1 Current Approaches

The idea that visual tracking can be considered a more general optimisation problem is not new and has been explored extensively in visual tracking literature. In this section the aim is to investigate several tracking systems that use traditional optimisation algorithms in visual tracking. The aim is to compare different objective functions and target models to find the best combination for use with the harmony search algorithm.

By efficiently combining a number of methods Nummiaro et al. designed a fast and robust tracking system also based on the particle filter [54]. Even though this method does not use a traditional optimisation algorithm it is of interest due to its target model. It uses colour histograms as a target model and formulates the tracking problem in such a way that gradient based optimisation algorithms can be used to perform target localisation. The *Bhattacharyya coefficient* is used to measure similarity between histograms and acts as the fitness metric when optimisation algorithms are used [81]. It is defined as

$$\mathcal{B}(t,c) = \sum_{i=1}^{N} \sqrt{t(i)c(i)} \ , \tag{3.55}$$

where N is the number of bins in the histograms and t and c are the histograms being compared. Notice that $\mathcal{B}(t,c)$ is large when the histograms are similar and small when they are very different. It is usually normalised so that a value between 0 and 1 is given with 1 denoting identical histograms. It was shown that this approach gives superior performance during large scale changes in the target or during rapid and erratic movement. Notice that the Bhattacharyya coefficient can be used as the basis for an objective function when colour histograms are used to model the target.

The use of colour histograms as a target model in tracking applications has been explored by many researchers. Chen et al. introduced the *Adaptive Color Vision System* (ACVS) that uses adaptive colour histogram models that automatically update themselves to adapt to changing light conditions [82]. The models adapt by learning a distribution of colours through various light conditions and using this distribution rather than the original histogram as the target model. The histogram back projection method, used to robustly recognise objects in an image, is then used to match targets with models [83]. This method works well in maintaining robust target acquisition through changing light conditions but fails when similarly coloured objects to the target enter the frame. This is due to the target model adapting to match objects from a range of similar colours in order to be robust during lighting changes.

Comaniciu et al. proposed a visual tracking method that uses colour histograms and formulates the problem so that gradient based optimisation algorithms can be used to perform target localisation [84]. Their method also uses the Bhattacharyya coefficient to measure the similarity between histograms. It creates a smooth objective function that is then optimised using the mean shift procedure. Various optimisations regarding the computational complexity of the algorithm is made resulting in a system that is able to perform real-time tracking robustly in environments containing visual clutter. This method proved to be robust under partial occlusion and was able to maintain target localisation during erratic motion and changing light conditions [84]. This approach again shows that colour histograms contain enough information for robust target localisation and that the Bhattacharyya coefficient is an efficient metric with which to formulate visual tracking as an optimisation problem.

These and other researchers showed that visual tracking can be achieved efficiently using gradient based optimisation algorithms. This inspired other researchers to investigate other optimisation algorithms that are not gradient based like the genetic algorithm and the particle swarm optimisation (PSO) algorithm.

Sulistijono et al. proposed a method using PSO for a head tracking application [8]. This is a completely different approach compared to the ones previously mentioned and does not use colour histograms to represent targets. Instead, template matching with a previously constructed head template is used to define the objective function. PSO or a genetic algorithm is then used to optimise the objective function to correctly localise the head tracker. The head template is simply an image of a generic head

indicating where hair and skin coloured pixels are expected to be found. Performance comparisons between the PSO and genetic algorithm implementations show that the genetic algorithm tends to converge to a more stable solution but that PSO is faster. Both algorithms can be adjusted to favour speed over accuracy by limiting the number of function evaluations until convergence.

Zang et al. also showed that the PSO algorithm can be successfully applied to visual tracking [9]. In their work the PSO algorithm is adapted specifically for use in visual tracking. The parameters that control the movement of the particles in the swarm is updated dynamically depending on the fitness values of the particles. In this way the temporal continuity information of the sequence is taken into account resulting in improved accuracy and robustness. It was also shown theoretically that this framework could be interpreted as a multi layer importance sampling based particle filter. Experimental results showed that this PSO based tracking system is robust and effective especially during fast and erratic motion.

Minami et al. used a genetic algorithm to design a visual servoing system that is able to recognise and track a target, in this case a fish, using only a grey-scale image [6]. They employ an elitist selection strategy in the genetic algorithm that maintains the best individual solution throughout all generations. This is similar to the strategy of PSO that also constantly keeps track of the best solution in the population. A genetic algorithm is used to minimise a function that measures the difference between a previously defined fish template and the image captured by the camera. This objective function is not related to the Bhattacharyya coefficient but is similar to the template matching approach used by Sulistijono et al [8]. This genetic-algorithm-based system proved successful in recognising and robustly tracking a fish target in real-time using limited amounts of computational resources.

Thus far we investigated three methods for visual tracking that use colour histograms as a target model. Two of them use the Bhattacharyya coefficient as a distance measure between histograms. It was shown that fast and robust visual tracking was possible using this combination of methods. The question that remains, however, is how to best optimise the objective function resulting from comparing histograms using the Bhattacharyya coefficient. This optimisation is central to quickly and accurately localising the target in the frame.

Fast and accurate optimisation of the objective function is an important aspect in all tracking systems of this type and the last two methods we investigated both used an evolutionary algorithm for optimisation. In both of these systems computational resources were limited yet real-time performance was achieved. This is due to both PSO and genetic algorithms being able to sacrifice accuracy for speed by forcing an early convergence in the algorithm by limiting the number of generations or particles. This trade off between speed and accuracy is a common feature of most evolutionary algorithms and allows a designer to adapt the algorithm to the size of the search space and the computational requirements. Notice also that even better results were obtained when evolutionary algorithms were adapted specifically to take advantage of the temporal continuity information in the visual tracking problem.

In the section that follows a novel method is proposed that combines the rich dataset found by modelling using colour histograms, with the flexibility of evolutionary optimisation algorithms like PSO and genetic algorithms. The *Harmony Filter* is introduced as an evolutionary algorithm based on harmony search for the visual tracking of an arbitrary target through a video sequence.

3.4 The Harmony Filter

The Harmony Filter (HF) is designed around the IHS algorithm and was developed as part of an investigation into the use of harmony search in computer vision [23–25]. Like many of the algorithms investigated in the previous section, the HF uses a colour histogram model to model the target. Localisation of the target is achieved by comparing histogram models of candidate localisations with that of the target. The model that represents the correct localisation will be most similar to the target model.

This similarity is measured using the Bhattacharyya coefficient (see Equation (3.55)). The Bhattacharyya coefficient measures the distance between two distributions or, as in this case, histograms. When this distance measure is minimised the two models that are being compared are the ones that are most similar. The harmony filter finds the most optimal localisation by using IHS to minimise the Bhattacharyya coefficient between the target histogram model and a histogram model from a candidate localisation. The Bhattacharyya coefficient is therefore the objective function in this optimisation problem.

Continual tracking is then achieved by first making a template colour histogram model of the target and saving it for future comparisons. In each frame of the video the target is localised by finding the localisation whose histogram model minimises the Bhattacharyya coefficient when compared with the saved template model.

In this way visual tracking, or the problem of locating the target in every frame, becomes a frame-by-frame optimisation problem. The objective function that measures the relative fitness of hypotheses (candidate localisations) in the solution space (the set of all possible localisations) is the distance between the candidate histogram and the target histogram. This distance is measured using the Bhattacharyya coefficient. The larger the Bhattacharyya coefficient, the more the candidate resembles the target and the better hypotheses the candidate represents.

The architecture of the harmony filter consists of two components. The main tracker part contains the user interface and has a frame grabber that receives a constant stream of images from a camera or video file input. It is also responsible for keeping a reference histogram of the target and displaying each frame in real-time once the target had been located. The second and novel part of the system is the Harmony-Search-based optimiser. It is responsible for locating the target in the image frame. This overview is illustrated in Figure 3.5.

In the sections that follow we investigate and discuss each component of the architecture outlined in Figure 3.5. The process starts with the user selecting the target by drawing a box around it. The tracker then creates a colour histogram of the area specified as the target and saves it as the template histogram. Once the template histogram has been created it is sent to the optimiser and saved as part of the objective function. Now the main loop that processes every frame on a per-frame basis starts.

The first step is to initialise the Harmony Search optimiser with the previous location of the target from the previous frame. The optimiser then finds and returns the target's current location and the frame is displayed with the target location marked by a square. This loop continues until no more frames are available.

3.4.1 Initialisation

The harmony search optimiser is initialised when the user chooses a target and its template histogram is generated. This is done by first converting the colour space from RGB to HSV. A two-dimensional histogram is then generated from the (H)ue and

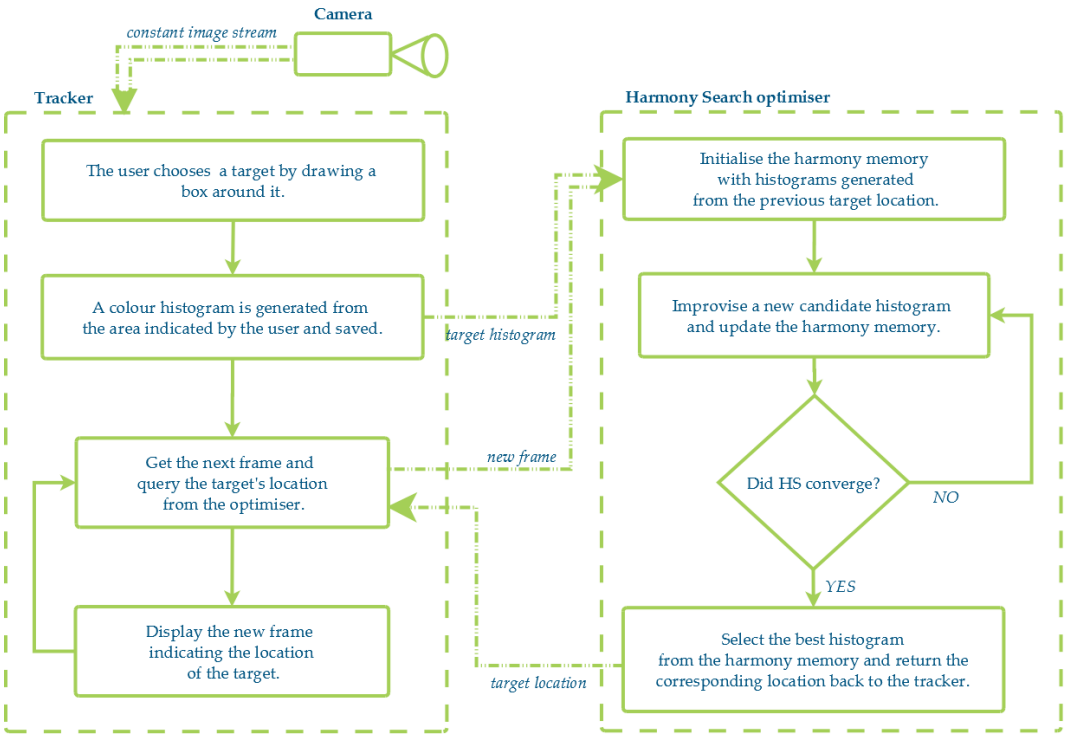

Figure 3.5: The harmony filter consists of two parts, namely the tracker and the harmony search optimiser. The tracker renders the optimal localisation of the target as an overlay on top of each video frame in real-time. It does this by using the harmony search optimiser to minimise an objective function that indicates the most likely localisation.

(S)aturation channels while the (V)alue channel is discarded. The V channel contains the intensity information and is sensitive to light changes [54, 85, 86]. The HSV colour circle is illustrated in Figure 3.6. By ignoring the V channel and concentrating on the H and S channels the model becomes more robust to changing light conditions between frames. By only using a two-dimensional histogram and not the full three-dimensional one one also gains an overall speed increase. This histogram is then saved by the optimiser as the template histogram and is used whenever a candidate histogram is evaluated.

The HSV colour system is not without faults and is not perceptually the most accurate representation of the hue, saturation and value components of an image [87]. However, HSV is a simple mathematical transformation of RGB making the transformation of the image to the HSV colour space computationally inexpensive. While not perfect the HSV colour space is accurate enough for the purpose of tracking and, more

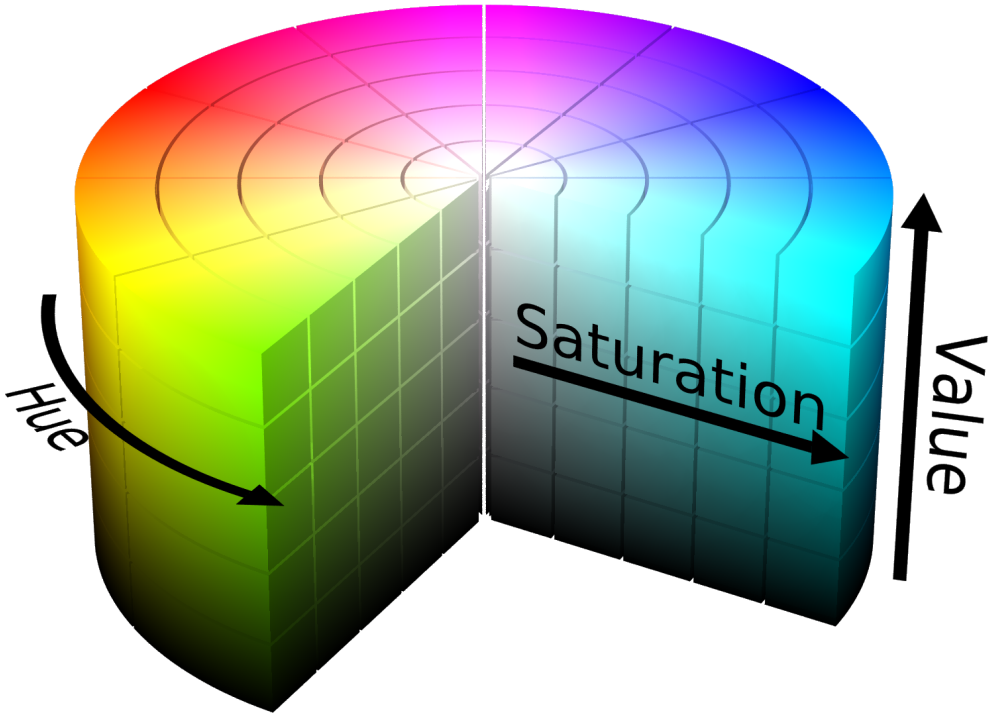

Figure 3.6: This diagram illustrates the HSV colour space by modelling it as a cylinder. The cylinder's height represents the V channel while the H and S channels are represented by the cylinder's circumference and radius respectively.

importantly, allows for real-time performance.

Colour histograms, also called colour distributions, are used as target model due to their robustness to partial occlusion, rotation and deformation. A histogram is a distribution of the colours that are present in the region of interest. Therefore, depending on how the region of interest is defined, the histogram changes very little when an object within the region of interest rotates or deforms.

The harmony filter uses a 2-dimensional histogram created from the H and S channels of the HSV colour space. The different H and S intensities are discretised into m and n bins respectively. This creates a 2-dimensional histogram with $m \times n$ bins. The choice of m and n is a trade-off between speed and accuracy. The more bins we use the more accurate we can represent the histogram but more bins also means makes the objective function more computationally expensive. It was found through experimen-

tation that 10 H bins ($m = 10$) and 12 S bins ($n = 12$) resulted in a good compromise that is both accurate enough for robust tracking without needlessly slowing down the calculation of the objective function.

However, the target histogram is only part of the initialisation. The template histogram only has to be calculated once and then never changes but other initialisation steps are performed at each new frame. Every time the optimiser is queried for the target location the harmony memory is initialised using the current frame and the target's previous location. The previous location is used to predict the state vector that describes the target's location, velocity, and scale as a five-dimensional vector. A state vector is defined as $\mathbf{x}_i = [x, y, \dot{x}, \dot{y}, s]$ where x, y is the target's location in pixel coordinates, $\dot{x}, \dot{y}$ is the target velocity, and s is a scaling parameter that controls the size of the box defining the target. Notice that the optimiser not only finds the target's most probable location but also its velocity and scale. However, once the scale and velocity has been estimated one can calculate the current location based on the previous location.

A simple motion model that assumes steady velocity of the target between frames is used to fill the HM with estimated predictions of the target location. A random acceleration in the x and y direction (a_x, a_y) is generated and used to create a new state vector as follows.

$$x_{t+1} = x_t + \dot{x}_t + \frac{1}{2}a_x \tag{3.56}$$

$$y_{t+1} = y_t + \dot{y}_t + \frac{1}{2}a_y \tag{3.57}$$

$$\dot{x}_{t+1} = \dot{x}_t + a_x \tag{3.58}$$

$$\dot{y}_{t+1} = \dot{y}_t + a_y \tag{3.59}$$

Each new candidate state vector is weighed by creating the corresponding histogram and comparing it with the template histogram using the Bhattacharyya coefficient. The new vector with its fitness weight is then added to the HM until the HM is filled.

Once the HM has been initialized new candidate solutions are improvised using the standard HS algorithm and the HM is updated until convergence to the optimal solution is detected. Since the predicted target position can be calculated from its velocity and previous position, only the $\dot{x}, \dot{y}$ and s components are explored during the

improvisation process. This speeds up the convergence by restricting the search space to only solution vectors that are possible within the motion model.

3.4.2 Convergence Detection

Convergence is detected using three separate tests. If any of the three tests indicate that the algorithm converged, the algorithm is terminated and the best solution vector found in the HM is returned as the optimal target position.

The first test measure the spatial distance (pixel distance in a frame) between the best solution in the HM and worst. If the distance is smaller than some threshold and the best solution has a sufficiently high fitness weight (determined by its histogram distance from the reference target histogram) the test passes and convergence is assumed. This test works by assuming that the optimiser has converged when all the solution vectors in the HM become nearly identical. This is usually a good indication that HS found the optimal solution but it is also possible that during the initialization phase all vectors get initialized to an area far from the true target position. Erratic target motion often causes a situation where all candidate solutions in the HM are equally bad but spatially close together indicating possible convergence. It is therefore necessary to test the weight of the best vector in the HM and ensure that it is sufficiently high to be confident that HS converged to the correct position before the search is terminated. Figure 3.7 illustrates how this test would detect convergence in three common situations.

When the first convergence test fails the second test counts the number of consecutive iterations that have not updated the HM. These idle iterations indicate that no progress is being made and that the target cannot be found or that convergence is slow. After a specified number of consecutive idle iterations it is assumed that no further progress will be made and the search is terminated.

When both the first and second test fails the final test bounds the search to a maximum number of iterations. If the number of iterations exceeds the maximum the search is terminated. The convergence tests prevent wasted computations by terminating the search early when no progress is being made or when they detect that the optimal solution has been found. It is important to keep wasted computations to a minimum to ensure real-time performance.

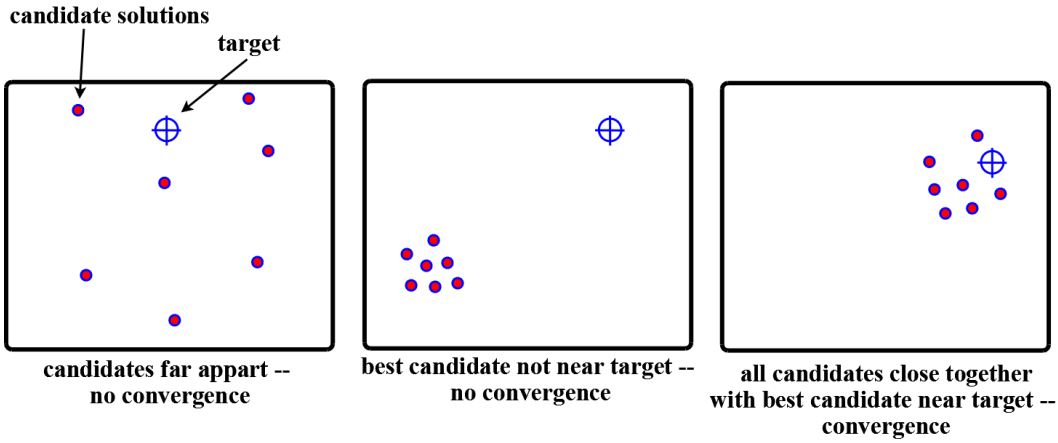

Figure 3.7: The first convergence test will fail if the candidates are spread out in the search space or if the best candidate histogram is not sufficiently similar to the reference target histogram. In the first example the candidates are too spread out and the search will continue. The second example shows incorrect convergence or bad initialization and the search will continue due to the best candidate not being similar enough to the target. In the last example the test passes and the search is terminated.

Notice that the first two tests are heuristic and that it cannot be guaranteed that convergence will always be accurately detected. By tweaking the three thresholds one can cause the algorithm to favour optimistic convergence detection or to favour pessimistic convergence detection. When it favours optimistic detection convergence is detected earlier and more often, so the algorithm becomes faster, but accuracy may suffer since further optimisation might have been possible. Pessimistic convergence detection has the opposite effect and will cause the algorithm to slow down since the maximum number of iterations are often used even though no further progress was made to justify the extra iterations.

I chose the first test's distance threshold small to favour accuracy over speed and found that a value of 3 pixels almost certainly implied true convergence when this test was positive for convergence. The fitness threshold guards against the unlikely event that all the initial improvisations are spatially close together but off the target. The value is highly dependent on the objective function which in this case is the Bhattacharrya coefficient. I chose the second threshold as 0.62 to assure that optimisation would continue if the best fitness is not at least equal to 0.62.

The second test's threshold is usually needed only when the tracker is lost and

is not converging to any single coordinate be it accurate or not. I choose its value to be one fifth of the maximum number of allowed iterations. So if the tracker is allowed a maximum of 500 iterations before being forcibly terminated, the second test terminates it after 100 idle iterations. It is unlikely that after 100 idle iterations any further progress will be made so in the interest of speed the search is abandoned and the *lost tracker* routine, explained in the next paragraph, is likely activated.

3.4.3 Lost Tracker Recovery

In challenging environments the tracker often loses the target momentarily. This is usually due to the target becoming partially or fully occluded by other objects in the frame or the target moving out of frame. An example of such a situation is seen in Figure 3.8. Erratic, unpredictable movement can also cause the tracker to become unable to find the target. When the tracker loses its target it must recover quickly to resume accurate tracking. First the optimiser is reset to search a larger area instead of concentrating on the area predicted by the motion model. Since the tracker is completely lost one expects that the prior information has become unreliable and therefore the search area is expanded.

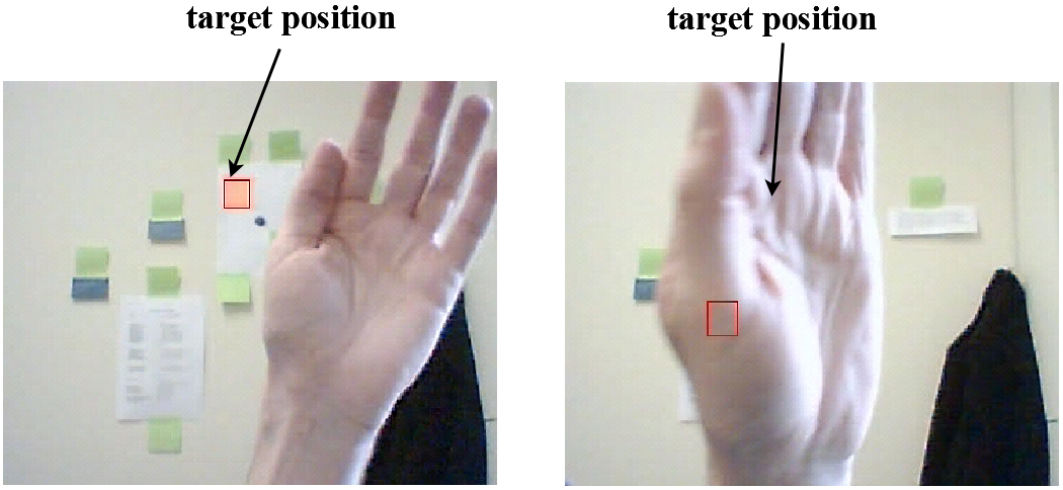

Figure 3.8: In this example the target is an orange square on the wall. In the left image the tracker is successfully tracking the target indicated by the red square drawn around the correct target position. In a later frame the hand occludes the target completely causing the tracker to lose the target.

Detection of a lost tracker is done by comparing the fitness weight of the best candidate from the previous frame with a specified threshold value. If the weight is below the threshold the tracker is considered lost and the search process is adapted until the best candidate's fitness is again above the threshold. Again, the choice of this threshold can be optimistic or pessimistic depending on the how much confidence there is on the target model and the tracking environment. The threshold should be chosen high enough so that a lost tracker can be detected as soon as possible to prevent multiple consecutive frames in which the tracker fails to find the target. However, if the threshold is chosen too high tracking quality will suffer. Through experimentation it was found that a threshold value of 0.5 ensured that the lost tracker routine is only activated when the tracker is really lost without wasting too many frames on determining this situation.

When the lost tracker routine is activated it adapts the search process by ignoring the motion model and initializing the HM with random solutions from the entire search space (the whole frame). The distributions used to generate random improvisations are also changed from normal around the motion model predicted solution, to uniform covering the whole search space. This indicates that the target can be anywhere in the frame and that all previous knowledge should be disregarded. For the same reason the target velocity is also reset to 0. In practice all this is accomplished by temporarily setting the HMCR= 0 until the best solution's fitness improves enough to rise above the 0.5 threshold.

The ability to quickly recover when the target is lost is one of the advantages that the Harmony Filter has over other popular tracking methods like the Kalman Filter. While both the Kalman and particle also have some ability to recover, the HF does this quicker and more efficiently as we will see in Section 3.4.8. Often when the target moves out of frame or becomes occluded the motion model, which plays a larger role in the Kalman and particle filters, will lead the tracker away from the target's true position. The tracker then becomes lost and will likely never recover if it only relies on its motion model for direction. The Harmony Filter generally recovers much more quickly due to its weak reliance on a predictive motion model.

In Section 3.4.8 the Harmony Filter is compared with a particle filter and a UKF based tracking system in challenging conditions. This ability to quickly recover from

occlusion and erratic movement by using a lost tracker routine is shown to be a main reason for its increased overall accuracy and performance.

3.4.4 Implementation

Pseudo code for the Harmony Filter is given in Algorithm .2. In this section a tutorial, based on algorithms .1, .3 and .2, is given on the implementation of the harmony filter. The aim here is not to provide full implementation details but rather to give an overview of how the components that were discussed in the previous sections are combined into an algorithm. For a much more detailed description of the harmony filter see Appendix 6.3.

First the HM is initialised based on the best solution from the previous frame. If this is the first frame the initial target location with zero velocity and a scale of 1 is used instead. For each new candidate three random values are generated. The first is sampled from a uniform distribution and is used to generate the new scale. The second and third are random accelerations that are independently sampled from the same normal distribution. These random values are then used to generate the components of a solution vector based on equations 3.56 to 3.59.

A new histogram is then calculated based on the area defined by the the new candidate solution vector. Together with the template histogram it is then used to calculate the fitness weight using the Bhattacharrya coefficient. The fitness weight is then appended to the solution vector which completes the generation of one member of the initial HM. This process is then repeated until all the members of the HM have been initialised.

Now the main search loop is started. First the PAR and the FW is updated as explained in Section 2.6.2 and equations 2.34 to 2.36. Then the best member of the HM is checked for quality to detect whether the tracker is lost (see Section 3.4.3). If the tracker is considered lost the HMCR is then set to zero.

The next step improvises a new candidate solution using the harmony filter improvisation step. The details of this step are found in Algorithm .3. The main difference between this step and the procedure explained in Section 2.2.1 is the use of the motion model of Equations 3.56 to 3.59. Only the velocity and scale is therefore improvised since the other components can then be implicitly calculated. All that then remains is to calculate the new candidate's fitness score and add it to the solution vector.

After improvisation the HM is updated if necessary. Each time the HM is updated the members are re-evaluated to determine which is the best and worst member currently in the HM. This determines both the current optimal solution and the member that currently qualifies for replacement when the HM is updated.

All that remains then is the convergence test. The procedure used for convergence testing was explained in Section 3.4.2 and more specific details are found in Algorithm .1.

The convergence test is the final step in the main search loop of the harmony filter. Once the main loop terminates, either because convergence was detected or because the maximum iterations has been reached, the best candidate of the HM is selected. This candidate is then returned by the harmony filter optimiser as the best solution. The target location is then extracted from this solution vector and is used to render a box on top of the current frame to indicate the current position of the target.

3.4.5 Setting the parameters

In Section 2.5.3 we analysed what the effect of certain parameters was on the performance of harmony search. In that section the focus was on optimally choosing the HMCR and the HMS. The PAR is another important parameter that can have a large impact on the performance of an algorithm based on harmony search. The analysis of the PAR was deferred to this section because, as we saw in the rest of Chapter 2, using a single constant PAR is far from optimal. With the harmony filter the IHS approach was used instead. Therefore, the analysis here should rather concentrate on the optimal values of PAR_{min} and PAR_{max} instead of PAR.

I will again concentrate on computational performance measured as the number of iterations until convergence, instead of accuracy. As we saw in Section 2.5.3 the effect that the harmony search parameters have on accuracy is minimal in the computer vision applications that are the focus of this study. We therefore test a range of PAR_{min} and PAR_{max} values to find the set that results in the least number of iterations until convergence.

In this experiment PAR_{min} is varied from 0.02 to 0.5 while PAR_{max} is varied from 0.5 to 0.98. As claimed by Mahdavi et al. [13], the effect of varying the PAR_{min} and the PAR_{max} is far less than changing the PAR directly and the algorithm therefore requires less fine tuning. A contour plot comparing the average number of generations

until convergence for various values of PAR_{min} and the PAR_{max} is shown in Figure 3.9. The overall optimal values were found to be $\text{PAR}_{max} = 0.56$ and $\text{PAR}_{min} = 0.14$ but the performance difference between these values and the values that did second best are almost negligible.

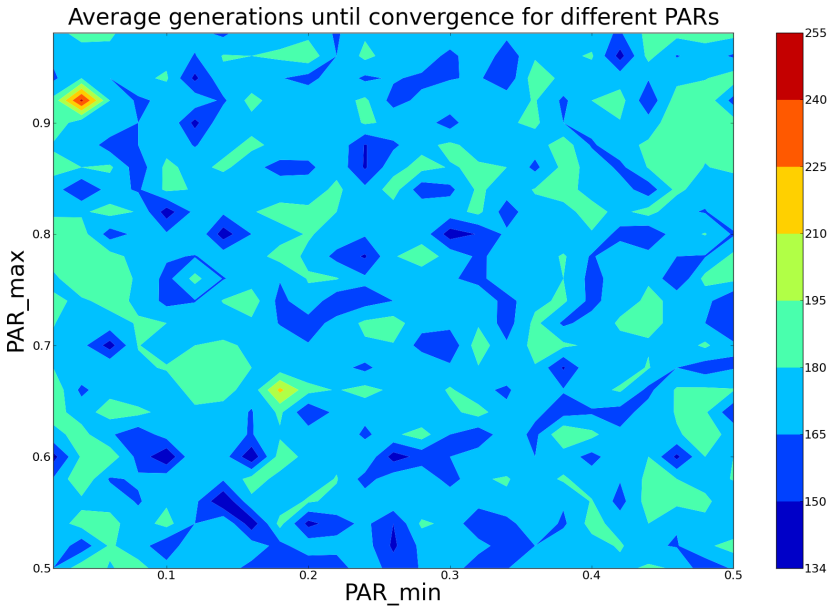

Figure 3.9: This contour plot shows the result that a range of values for the PAR_{min} and the PAR_{max} parameter has on the speed of the harmony filter. The average number of iterations until convergence is shown for various values.

Other important parameters of the harmony filter include the thresholds for convergence detection, lost tracker detection and the noise variances used to generate random scales and velocities. The choice for most of these values have already been discussed in sections 3.4.2 and 3.4.3. However the choice of the noise variances was not discussed.

Both the velocity noise variance and the scale noise variance should be chosen based on the size of the video frame and expected movement of the target. For example consider surveillance footage from a security camera in a shopping mall. Let the video capture frames of 500×500 pixels and the primary targets for tracking be people walking by the camera. If video is captured at 30 frames per second and the area in the frame represents 5 meters of a corridor, we can calculate the number of pixels that a target is expected to move based on the average walking speed of people.

The average walking speed of adults has been calculated to be 80 meters per minute [88]. If each video frame represents 1/30th of a second we therefore expect a human target to move 44.4 millimetres each frame. If 500 pixels represent 5 meters of corridor each pixel of the frame then represents 10 millimetres. Therefore, even if a target was moving at twice the average speed we would not expect it to move more than 15 pixels per frame. For this application a reasonable choice for the velocity noise variance is therefore 15.

The scale noise variance is more difficult to calculate and is usually determined experimentally. In my experiments a reasonable assumption was that a target would never grow by more than 50% between frames so the scale noise variance was set at 1.5.

Notice that all the experiments of this section as well as those of Section 2.5.3 were made under the assumption that all the parameters are uncorrelated and that optimisation of harmony search parameters can be done independently from one another. This assumption was made for practical reasons and may not be reasonable but this aspect has not been explored and further research is required before a theoretically strict optimisation can be done.

3.4.6 Computational complexity

The primary loop of the harmony filter is the improvise-and-update loop that is repeated once every iteration. Only one evaluation of the objective function is done in each iteration so the time spent optimising is almost fully dependent on the number of iterations until convergence. The number of iterations needed for convergence cannot easily be determined since it is dependent on the state of the target and its surrounding environment at that time. This also means that the time until convergence will be different for each frame and that the tracker's frame rate will not be constant.

An example of this inconsistent convergence speed is shown in Figure 3.10. It shows the number of iterations needed for convergence at every frame in a challenging tracking example that will be investigated further in Section 3.4.8. Notice that the tracker converges quickly at the beginning of the sequence but later frequently only converges after 500 generations which was set to be the maximum number allowed in this example. This is due to the target starting in an area where it is relatively easy to track and then later moving away from the camera to an area where it is often occluded

and where there are more visual distractors. Moving away from the camera also causes the target to become smaller which means a smaller histogram is created and less data is available to identify the target. As the conditions for accurate tracking become worse the number of iterations needed to converge increases.

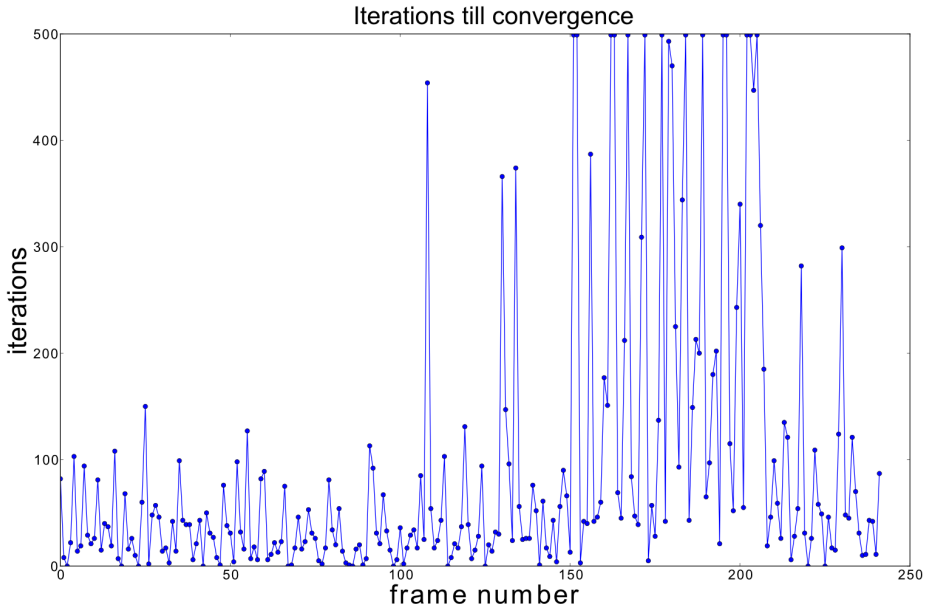

Figure 3.10: The graph illustrates the inconsistent number of iterations required for convergence that is typical of the harmony filter. In many frames less than 100 iterations are required but under more challenging conditions the maximum number of 500 is reached.

It is therefore important that the criteria for convergence are chosen carefully so that the number of iterations until convergence is kept to a minimum. We know that three tests are used to detect convergence and this example illustrates all three. When the target is easily identified, as was seen in the first few frames, convergence is usually detected with the first test since all the candidates in the HM quickly converge to the same solution. This test often correctly detects convergence in less than 100 iterations as seen in Figure 3.10. However, when tracking becomes more difficult all the candidates will not converge to the same point and the first test will not be able to detect convergence. This is seen later in the example when the target starts moving away from the camera. If the target is occluded or has moved out of frame the tracker will not find any good solution and the second test will eventually detect this state and

converge due to no progress being made for a prolonged number of iterations. This always leads to slower convergence since the algorithm will only decide that no further progress can be made after numerous idle iterations. In the example of Figure 3.10 the maximum number of idle iterations was set at 100 and we often see points in the graph where convergence took more than 300 iterations. These points are likely examples of frames in which the search was abandoned due to reaching the maximum number of idle iterations.

However, when multiple local distractors are present it may appear to the algorithm that it is slowly making progress by converging to multiple weaker local optima. This is the worst possible situation and it leads to the maximum number of iterations reached before the search is terminated. We see this happening a few times in the example of Figure 3.10 when 500 iterations are needed for convergence.

It is clear that accurately detecting convergence is the key to minimising the computational complexity of the harmony filter algorithm. However, when the tracker is lost and cannot find any good solutions it is better to continue on with the next frame in the hope that the situation improves rather than wastefully searching for a solution that does not exist.

Consider the example of Figure 3.11. It illustrates the evolution of the HM by plotting the weight of the best candidate for every iteration. First, notice that the initial best weight is less than 0.36 and the final weight is less than 0.48. These low weights indicate that the tracker is clearly lost and that even after 500 iterations of searching the target could still not be found. The target is most likely occluded and the best strategy would be to continue on to the next frame.

According to the graph the search should have been abandoned after about 120 iterations since no further progress to the best hypothesis was made. However, this is not as simple as it seems. Consider the more detailed view of the HM shown in Figure 3.12. It shows the pixel location of both the best and worst hypothesis in the HM as pixel coordinates. Notice that the smallest weight hypothesis changes frequently and is still changing after 450 iterations. This indicates that the HM was still being optimised even thought the final result, the best hypothesis, reached its final value after less than 100 generations.

Contrast this result with the example of figures 3.13 and 3.14. In this case the HM is initialised with good hypotheses indicated by the high initial weight. After 207

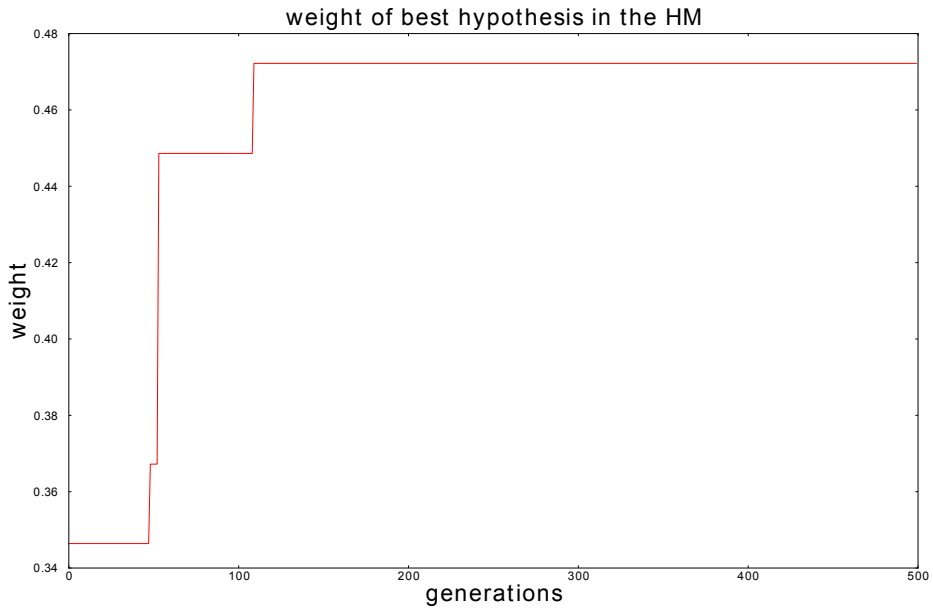

Figure 3.11: In this example the tracker is lost but continues to search until the maximum number of iterations is reached. This is not the desired behaviour and it would be better to abandon the search much earlier.

iterations the HM converges to a solution with a weight higher than 0.7 indicating that the target was most likely identified correctly. Notice also from Figure 3.14 that the final location of the best and worst hypotheses are less than 5 pixels away from each other indicating that it was the first convergence test that correctly detected convergence here.

Deciding when to stop the search process and continue on to the next frame is a trade-off between speed and accuracy. The harmony filter would benefit greatly from a better way of detecting occlusion and when there is no possibility of further improving the hypotheses in the HM. The overall frame rate of the tracker could be significantly improved if better convergence tests were provided but this is a topic of future research.

3.4.7 Theoretical advantages

As we will see in the results that follow in the next section, the harmony filter outperforms many modern approaches to visual tracking including the particle filter. It

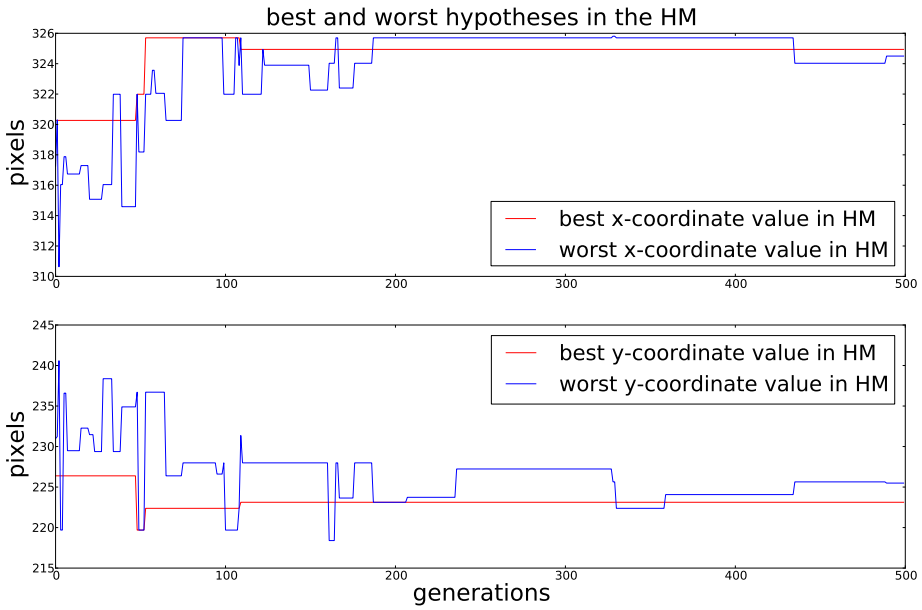

Figure 3.12: These graphs gives a more detailed view of the situation in Figure 3.11 and shows why convergence was not detected. The HM is constantly updated even though the best solution never improves.

is therefore appropriate that we investigate the theoretical advantages of the harmony filter as a possible explanation of the superior performance. The investigation will focus on the way prior information about the target motion is combined with the most recent observation. This is compared with the way the particle filter does this and I argue that the harmony filter makes better use of all available information.

In most particle filters the motion model that determines the prior information is used as the *proposal distribution function* from which the initial particle cloud is drawn. These particles are sampled from a distribution function that does not take observational information from the most recent frame into account and is therefore frequently inaccurate. This inaccuracy is especially bad when fast erratic motion causes the motion model to fail in accurately predicting the target's location. However, the particle filter does not ignore observational information but uses it to score the particles by their probability of having located the target. A *resampling* step is then performed that filters out low probability particles and duplicates high probability ones.

Over several frames this process guides the particle cloud to the most likely location

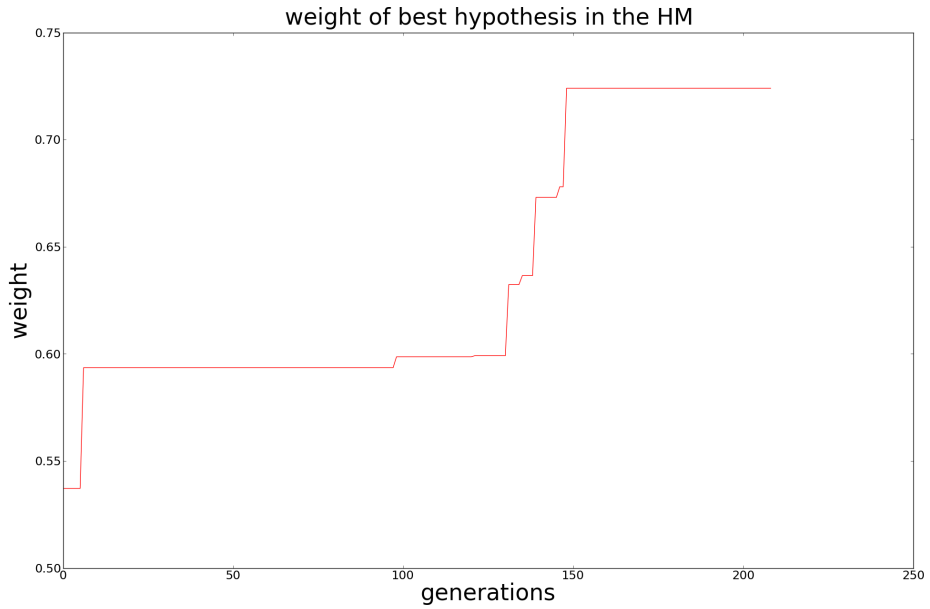

Figure 3.13: In this example all the candidates quickly converge to the same point and convergence is correctly detected.

of the target. However, notice that the location of the particles in the search space have not changed since the initial sampling from the proposal distribution function. This means that inaccuracies from the motion model can only be corrected, at best, in the next frame following resampling.

In contrast, the harmony filter uses the most recent observation extensively and uses prior information only to initialise the search process. As mentioned in previous sections, the harmony memory is initialised by propagating the previous target location through the motion model. This is similar to the way the particle filter generates the particle cloud by propagating the resampled cloud from the previous frame through the motion model. However, unlike the particles of the particle filter, the improvisations in the harmony filter change their position in the search space many times during the optimisation process by comparing with current observational information. In this way inaccuracies in prior information from the motion model can be immediately corrected and the error does not propagate to the next frame. Notice though, that when the prior information is accurate the harmony filter makes full of this by detecting early

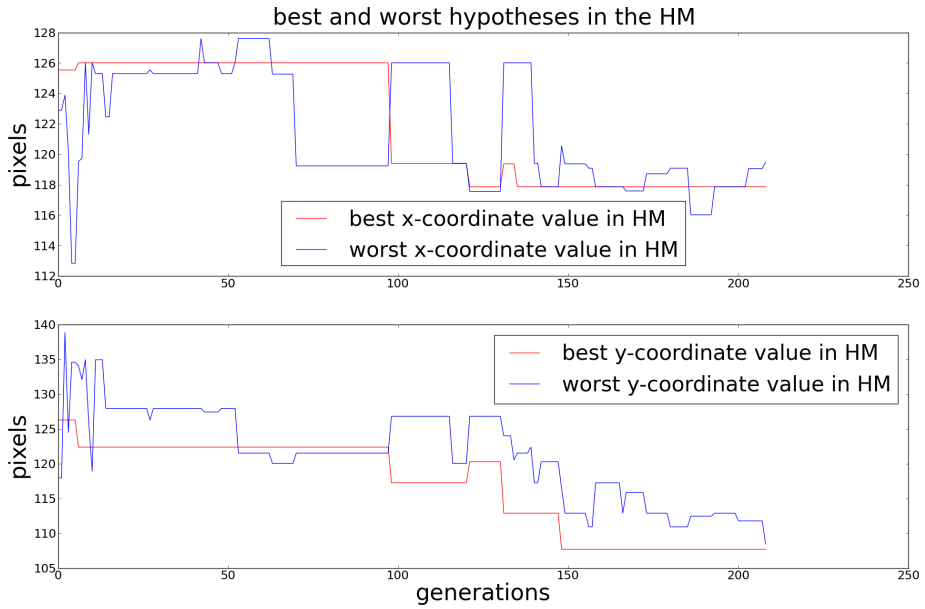

Figure 3.14: In this detailed view we see that convergence was detected correctly since the location of the best and worst candidates converged to the same point.

convergence and saving computational resources by terminating the search process. In contrast the particle filter always evaluates all particles in every frame even when an earlier evaluated particle successfully located the target.

By effectively using prior information when it is accurate and discarding it when it is not, the harmony filter potentially gains a theoretical advantage over the particle filter that always relies on prior information to generate the particle cloud. The harmony filter relies heavily on the most current observational information to improve the initial improvisations in contrast with the particle filter that only uses it to score particles without changing their initial position in the search space.

3.4.8 Results

In the final section of this chapter I demonstrate the harmony filter's ability to track objects in various challenging environments. The harmony filter is also compared with various other successful tracking algorithms including the unscented Kalman filter (UKF) and the particle filter. I concentrate on three video sequences that test the tracker's

ability to accurately track a target through quick erratic motion, local distractors, occlusion, low resolution images, bad and variable lighting and the target moving in and out of frame.

In all three examples I use the parameter values found by the sensitivity analysis of Section 2.5.3 and 3.4.5. FW_{min} and FW_{max} were chosen as 1 and 12 respectively to favour accuracy over speed. The threshold values for the convergence test were chosen as detailed in Section 3.4.2. To ensure a fair comparison none of these values were modified or fine tuned to specifically improve performance in any of the three experiments but was kept at the values stated here. In order to ensure that the comparison with the UKF and particle filter implementations is fair I use the exact same motion model discussed in Section 3.4.1, as the dynamic model used in updating the particle filter and the UKF.

The first two test sequences were captured with a low cost web cam at a resolution of 352×288 at 15 frames/second. The first example was designed to test the robustness of the harmony filter under very challenging conditions and thus includes all the tracking challenges associated with occlusion, poor quality images, changing light and local distractors.

In this test sequence a man walks and runs along a wet footpath. A web cam attached to a window of a neighbouring building records the man's movements from an angle that makes the man's bounding box appear to grow and shrink as he moves closer to and further away from the camera. The path is highly reflective due to it being wet and is partially occluded by trees. A few frames from this sequence (without a bounding box) is shown in Figure 3.15.

The sequence starts with the target standing still for several frames at one end of the path and then quickly running in a zigzag way to the end of the path. Near the end of the path the target becomes partially occluded by leaves from the tree in the foreground. He then changes direction and runs back the way he came. The transition from standing motionless to quickly running combined with rapid changes in direction from the zigzag motion cause the motion model to fail in accurately predicting the target's next location. This often causes the tracker to lose the target which makes this example highly appropriate for testing the tracker's ability to recover in this situation.

In order to test tracker accuracy the video was manually labelled by identifying the centre of the tracked object in each frame visually. The distance between the

true centre and the centre estimated by the tracker is then calculated and used as an accuracy metric. This only gives an indication of the true accuracy though since scale changes are ignored. Figures 3.16 and 3.17 show the speed and accuracy of the harmony filter compared with two different implementations of a particle filter based tracker (one using 300 particles and one using 500) and an unscented Kalman filter based tracker.

Notice from Figure 3.17 how the particle filter implementations were unable to recover from losing the target near the start of the sequence while the UKF and Harmony Filter managed to track the target for much longer. Near the end of the sequence the Harmony Filter loses the target before the UKF does but immediately starts recovering and eventually accurately recaptures that target. However, when the UKF loses the target it just drifts further away without ever recovering. This is likely due to local distractors pulling the particle clouds of the particle filters away from the target and the fairly predictable target motion causing the UKF to accurately predict the target's next position. The harmony filter proved robust in the vicinity of local distractors and could also quickly re-acquire the target after occlusion. This is something the UKF had trouble with but managed better by the particle filters.

In Figure 3.16 one notices that the performance of the Harmony Filter varies much more than it does with the other trackers. This is due to the convergence detection that might terminate the search early or late depending on its confidence in the solution. The particle filter implementations are clearly slower than both the UKF and the Harmony Filter but the difference in speed between the UKF and the Harmony Filter is negligible in this example with the Harmony Filter only being slightly faster on average. The average time until convergence for all three algorithms is shown in Table 3.1 and the average accuracy is shown in Table 3.2.

Algorithm	**Time until convergence (ms)**
Harmony Filter	20.44
UKF	24.22
Particle Filter (300 particles)	45.73
Particle Filter (500 particles)	72.48

Table 3.1: Average time (ms) until convergence for the running man test sequence.

Two examples that illustrate the difference in accuracy between implementations are shown in Figure 3.18. For each example three screen captures are shown that correspond

Algorithm	Average distance (pixels)
Harmony Filter	67.09
UKF	50.87
Particle Filter (300 particles)	167.11
Particle Filter (500 particles)	153.04

Table 3.2: Average distance (pixels) to target for the running man test sequence.

to the harmony filter, a particle filter implementation and a UKF implementation. In each image the estimated position of the target is shown by a red square.

In the second test sequence an orange square is tracked on a background with few local distractors. This environment makes it much easier for the tracker to recognise the target and one would expect the performance of all the trackers to improve compared to the previous test sequence. However, this video was constructed to test the tracker's performance during quick erratic movement that is difficult to predict using the simple motion model used in the harmony filter (which is also the same motion model used in the particle filter and UKF when comparing results). This motion was generated by rapidly rotating and translating the camera in random directions. The camera's lens was also occluded for prolonged periods of time to test the tracker's recovery of the target after it has been lost for many frames. Some frames from the video that illustrate this is shown in Figure 3.19.

The results from the performance comparison of the second sequence is seen in figures 3.20 and 3.21. In the speed comparison graph we see a situation similar to the one in the first example. While the particle filters and the UKF show fairly constant performance, the harmony filter's performance varies depending on the environment at that frame. Due to the prolonged periods of occlusion, during which the harmony filter took a long time to converge, its speed is more comparable to the other trackers but is still slightly faster on average. Table 3.3 shows the average time until convergence for this example and Table 3.4 shows the average accuracy.

Algorithm	Time until convergence (ms)
Harmony Filter	20.71
UKF	23.29
Particle Filter (300 particles)	39.52
Particle Filter (500 particles)	64.95

Table 3.3: Average time (ms) until convergence for the orange square test sequence.

Algorithm	Average distance (pixels)
Harmony Filter	48.47
UKF	150.30
Particle Filter (300 particles)	106.55
Particle Filter (500 particles)	71.07

Table 3.4: Average distance (pixels) to target for the orange square test sequence.

However, when comparing accuracies the harmony filter proves to be the most robust implementation. As soon as the erratic motion started the UKF lost track of the target and never got the opportunity to recover properly. The particle filters also lost track but sometimes recovered after only a few frames. The harmony filter recovered almost immediately from erratic motion and prolonged occlusion, and was still robustly tracking the target at a time when all other trackers were still trying to recover.

Two examples illustrating this claim are shown in Figure 3.22. As before three images are shown for each example, one for every filter implementation. In the first example the camera lens is covered by a hand for several frames and then removed. The harmony filter recovered after the hand was removed and immediately found the target again. However, the particle filter and the UKF could not recover from this situation. A similar situation is seen in the second example. Quick erratic motion causes the UKF and particle filter to lose the target without recovering but the harmony filter managed to only lose the target momentarily and quickly recovered.

The final test sequence that we will look at comes from a video recorded for the CAVIAR project [89]. It was chosen to test the tracker's ability to accurately track a target that changes its basic shape constantly. The sequence shows two men fighting each other and ends with one man knocking the other to the floor and running away. The men often occlude each other or part of each other, and by ducking, waving their arms and falling to the floor they change their shape as well. Notice also that the rectangle used to identify and model the target is now a very poor fit to the shape of the target. This makes it difficult for trackers to maintain the target position through local distractors in the scene and only the most robust ones will succeed. For these reasons it is challenging to track one of the men using a system that models targets as histograms of rectangular patches. Some frames from this test sequence is shown in Figure 3.23.

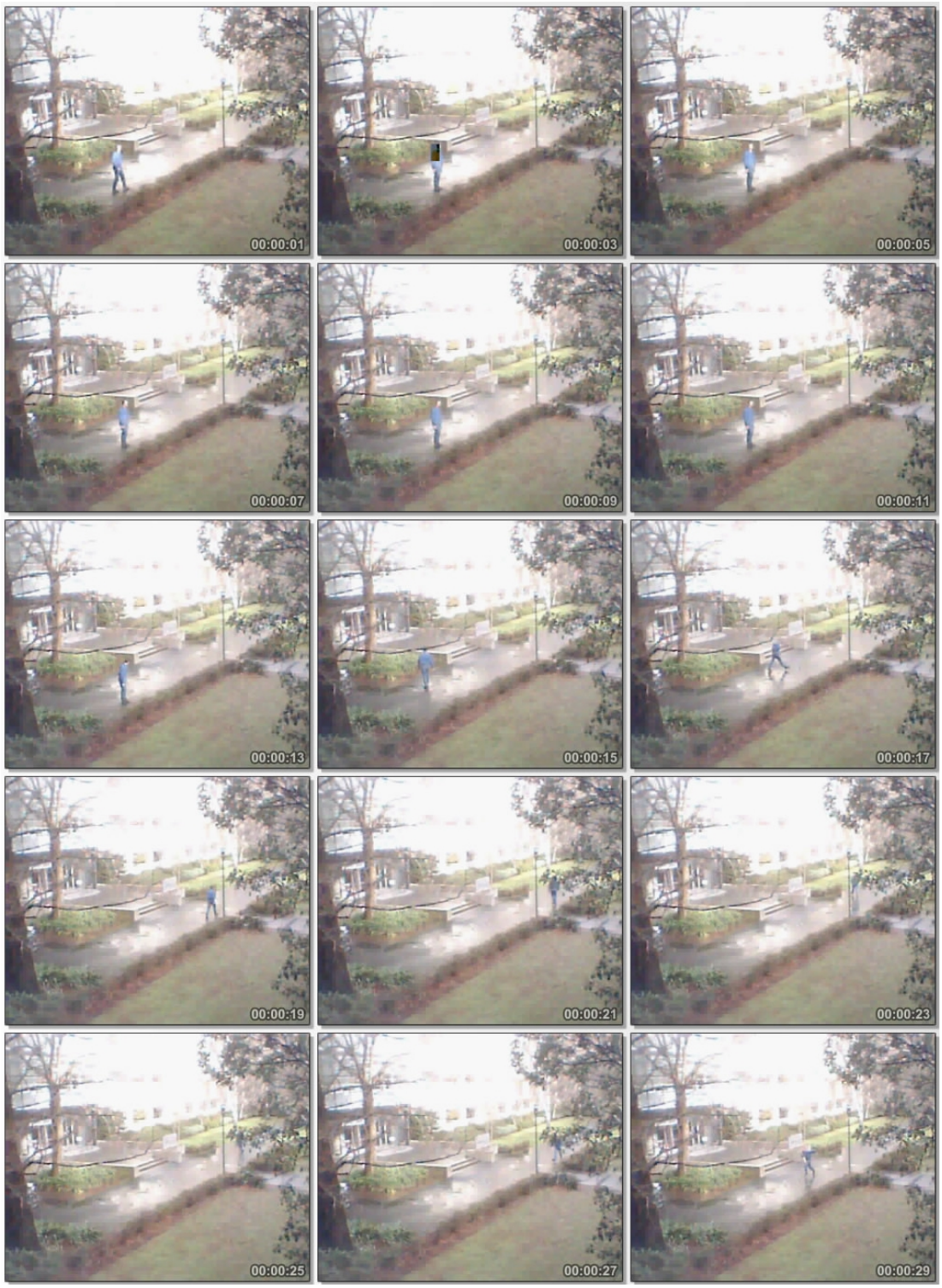

Figure 3.15: These are some frames from a low quality video sequence showing a man walking and running along a footpath. This is a challenging tracking problem due to various artefacts in the video and its low quality.

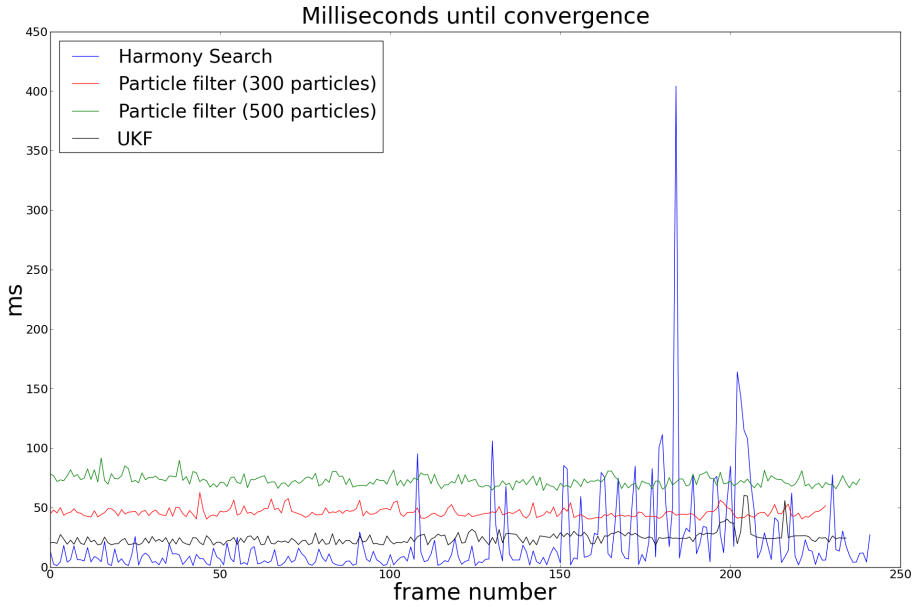

Figure 3.16: This graph compares the speed of various tracking algorithms when tracking a target under challenging conditions. It shows the milliseconds until convergence for each algorithm over all frames in the video sequence. The particle filter implementations lag behind the Harmony Filter and UKF implementations by at least 20ms while the performance difference between the harmony filter and the UKF is negligibly small for this example.

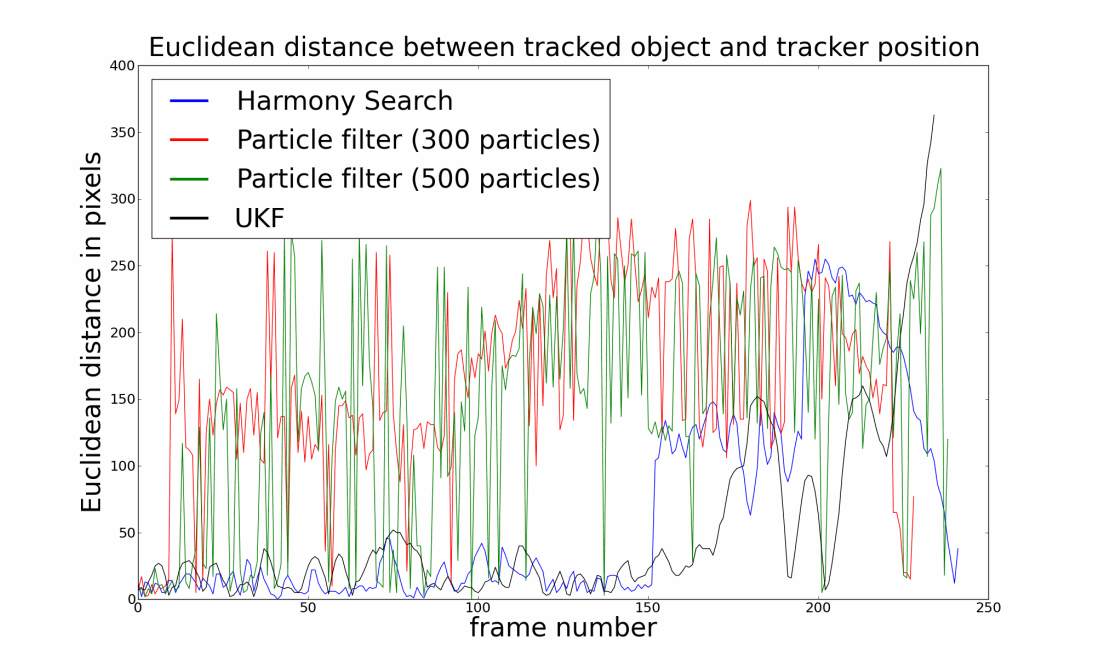

Figure 3.17: This graph compares the accuracy of various tracking algorithms when tracking a target under challenging conditions. It shows the distance in pixels between the true target centre and the tracker's estimate for each algorithm over all frames in the video sequence. In this example the particle filter implementations suffer from early target loss without recovery while both the UKF and Harmony Filter performs well until almost the end of the sequence. Both trackers eventually lose the target due to occlusion but the Harmony Filter eventually recovers while the UKF does not.

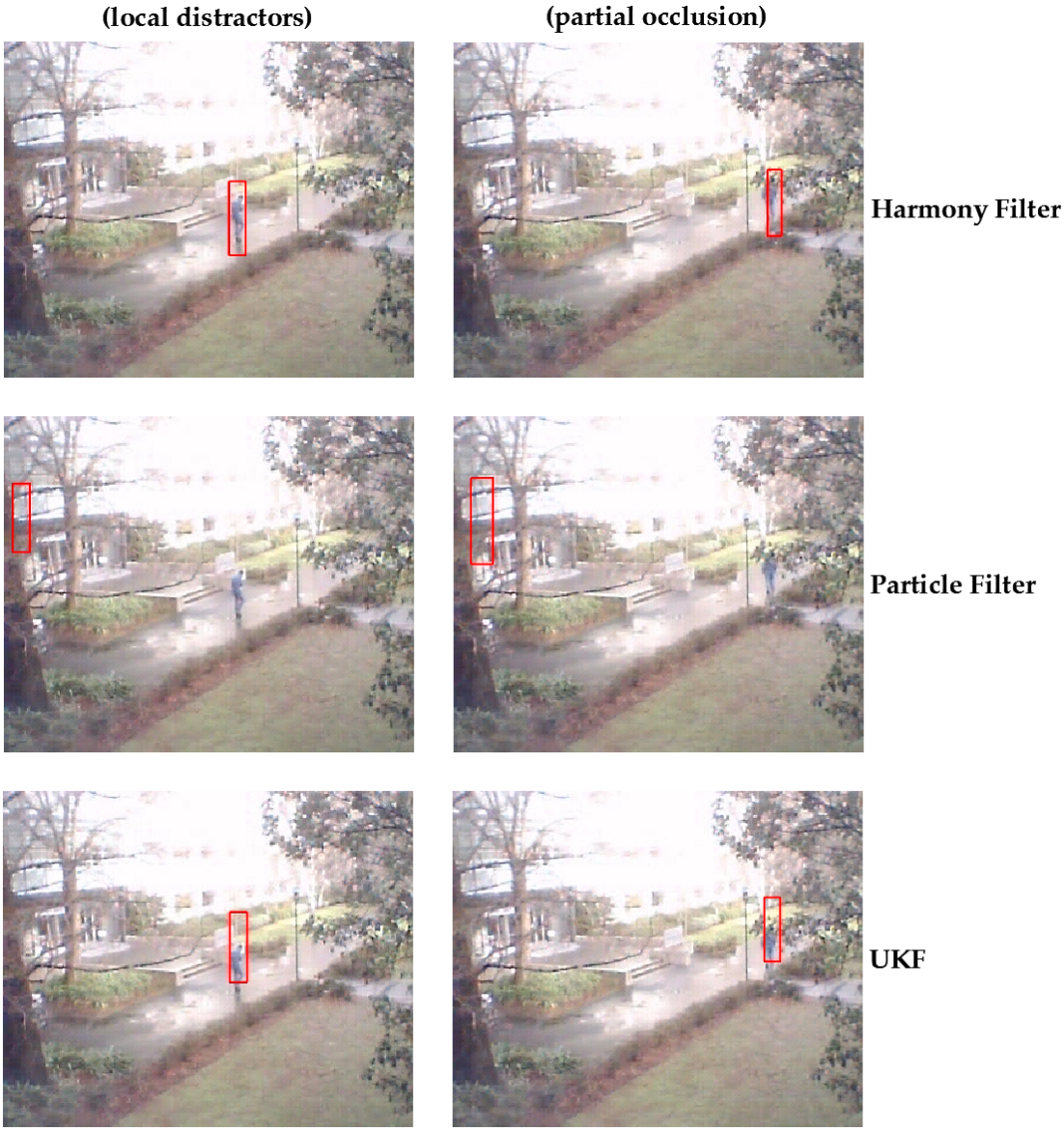

Figure 3.18: Illustrated here are two challenging tracking scenarios from the the first test sequence. In the first scenario (the first column of this figure) local distractors in the reflective pavement and changing light conditions cause the particle filter to get lost. The UKF and harmony filter proved to be more robust. In the second scenario the target is partially occluded causing the particle filter to again lose the target. Both the UKF and harmony filter managed to stay with the target but the harmony filter was better aligned with the true position of the target.

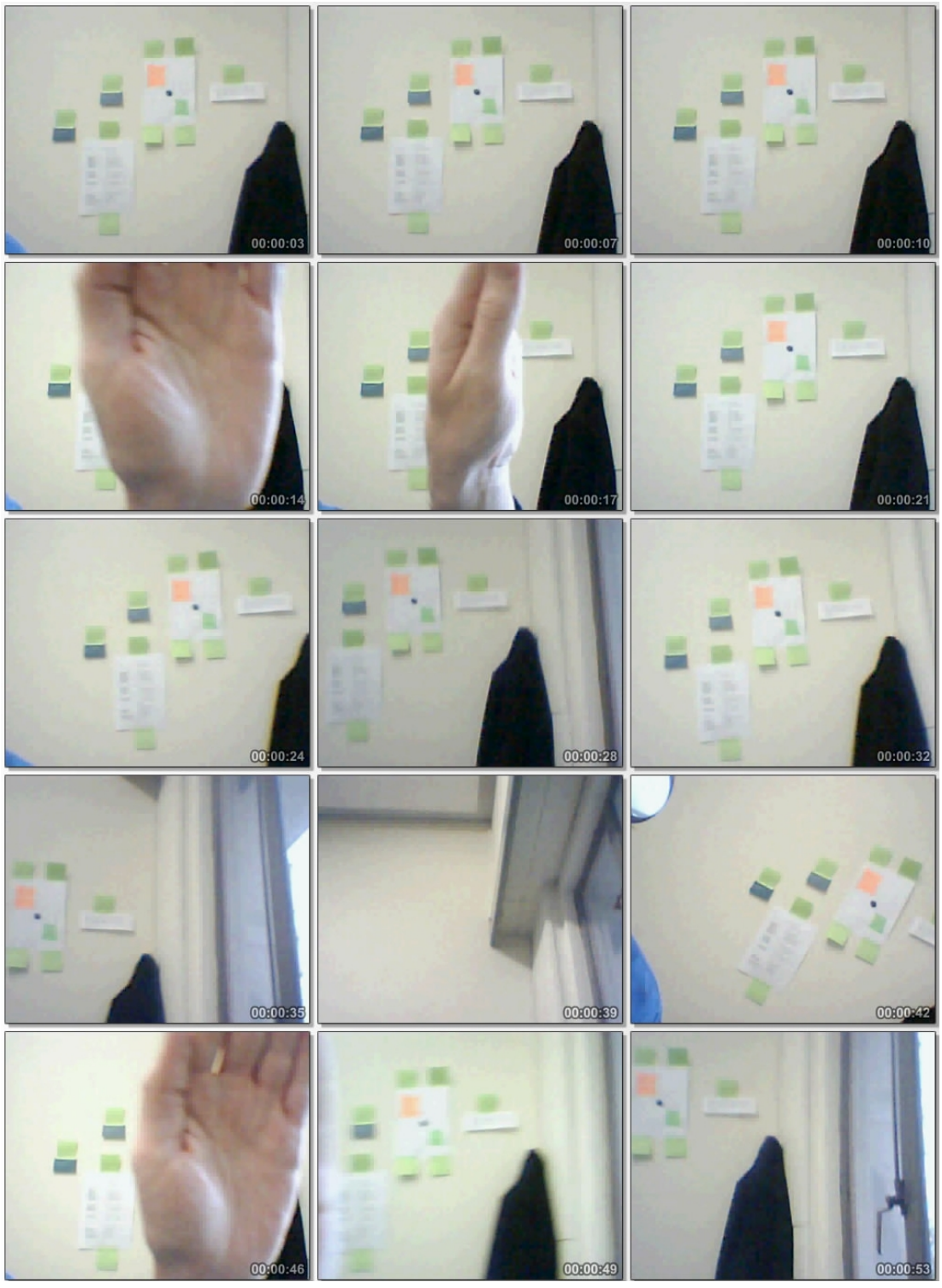

Figure 3.19: In the second test sequence an orange square is tracked in an environment that makes it easy for histogram based trackers to maintain the target. However, unpredictable and erratic motion combined with long periods of occlusion makes this a challenging problem.

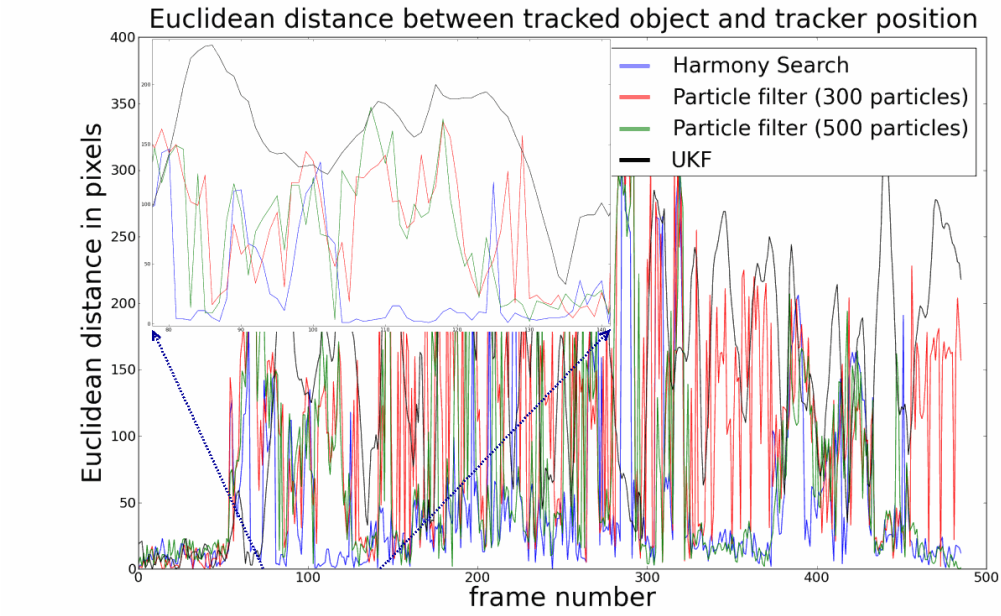

Figure 3.20: An accuracy comparison between different tracker implementations for the orange square test sequence. The sub-image is an enlarged section of the graph illustrating and comparing performance immediately following a long period of occlusion. The Harmony Filter is shown to recover quickly from occlusion while the UKF and particle filters only momentarily recover the target for one or two frames.

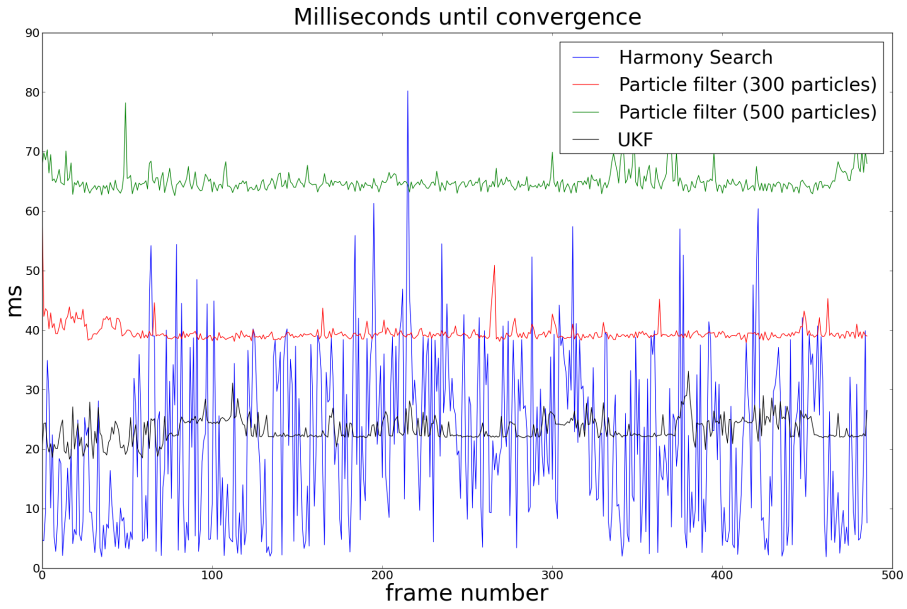

Figure 3.21: The speed of the harmony filter is compared to the other algorithms using the orange square test sequence. While the particle filters are again the slowest, the UKF and the harmony filter shows very similar performance on average in this example.

Figure 3.22: Two challenging tracking scenarios are shown from the second test sequence. In the first example the target is occluded for several frames and recovery from occlusion is compared between the various trackers. In the second example the camera is pulled and rotated to simulate quick and erratic motion of the target. Only the harmony filter could successfully track the target in both situations.

Figure 3.23: The third test sequence shows two men fighting. The man with the black shirt is the tracker's target.

Two tracking examples from this sequence are shown in Figure 3.24. The aim is to track the man wearing black and in the first example the man wearing the light blue shirt partially occluded the other man by moving in front of him. This action caused both the particle filter and the UKF to lose the target while the harmony filter managed to robustly track it. In the second example the man in black is knocked to the ground causing the model of the target to change considerably. This made all three trackers much more sensitive to local distractors in the image and only the harmony filter managed to stay with the original target.

Speed and accuracy comparisons from this example are shown in figures 3.25 and 3.26. The speed comparison was similar to the previous examples with the harmony filter again the fastest on average. It also proved to be the most accurate even though it lost the target near the end of the sequence without successfully recovering. The average times until convergence are shown in Table 3.5 and the average accuracy is shown in Table 3.6.

Algorithm	**Time until convergence (ms)**
Harmony Filter	11.66
UKF	25.45
Particle Filter (300 particles)	45.18
Particle Filter (500 particles)	69.61

Table 3.5: Average time until convergence for the fighting men sequence

Algorithm	**Average distance (pixels)**
Harmony Filter	38.98
UKF	92.59
Particle Filter (300 particles)	68.49
Particle Filter (500 particles)	65.50

Table 3.6: Average distance (pixels) to target for the fighting men sequence.

In the fighting men test sequence the tracker's accuracy was highly sensitive to the way the user identified the target. In other words, how large and where the user drew the box that defines the object to be tracked. When the box was drawn large enough to cover the entire man too much of the background was included in the calculation of the histogram and the target was poorly represented. Due to the target constantly changing shape without the tracker compensating for it, the background was frequently

identified as being part of the target. This is more a problem with the way the targets are represented than with the algorithms used to track them and is partially mitigated by choosing to track a smaller portion of the target instead of the whole target. In this example choosing to track just the torso of the man worked well and produced the best results for all the trackers. However, due to the target rotating and causing the torso area to become larger and smaller, the problem was still not completely eliminated and added to the difficulty of the problem. With this experiment I therefore show that problems caused by poor target representation can at least partially be compensated for in some situations using the harmony filter.

In this section three example tracking problems were used to evaluate the harmony filter's ability to robustly and efficiently track an arbitrary target through a video sequence. Speed and accuracy metrics were used to compare its performance with other popular algorithms used in visual tracking. The same target model and motion model was used in all tracker implementations. In all three examples the harmony filter proved to be consistently more accurate than the other algorithms and was also faster on average.

By carefully constructing and choosing the test sequences, I tested the harmony filter's performance under most of the conditions that commonly cause trackers to fail. In the first test sequence of the running man the tracker's robustness to poor quality video, local distractors, partial occlusion and poor target modelling was tested. The focus of the second sequence, the one with the orange square, was robustness under unpredictable and rapid erratic motion. Later in the sequence the tracker's ability to recover from long periods of total occlusion was tested. The final sequence was chosen for the target's large range of movements resulting in dramatic deformation of the original target shape. This causes the target model to frequently fail completely in accurately describing the target. The aim therefore was to see how much the tracking system was able to compensate and when it would fail completely.

The harmony filter proved to be robust in all these situations and was able to successfully track the target in real-time in most of the frames. In the concluding section that follows I discuss the harmony filter and its performance in the light of the results from this section as well as the way it compares with other popular tracking systems.

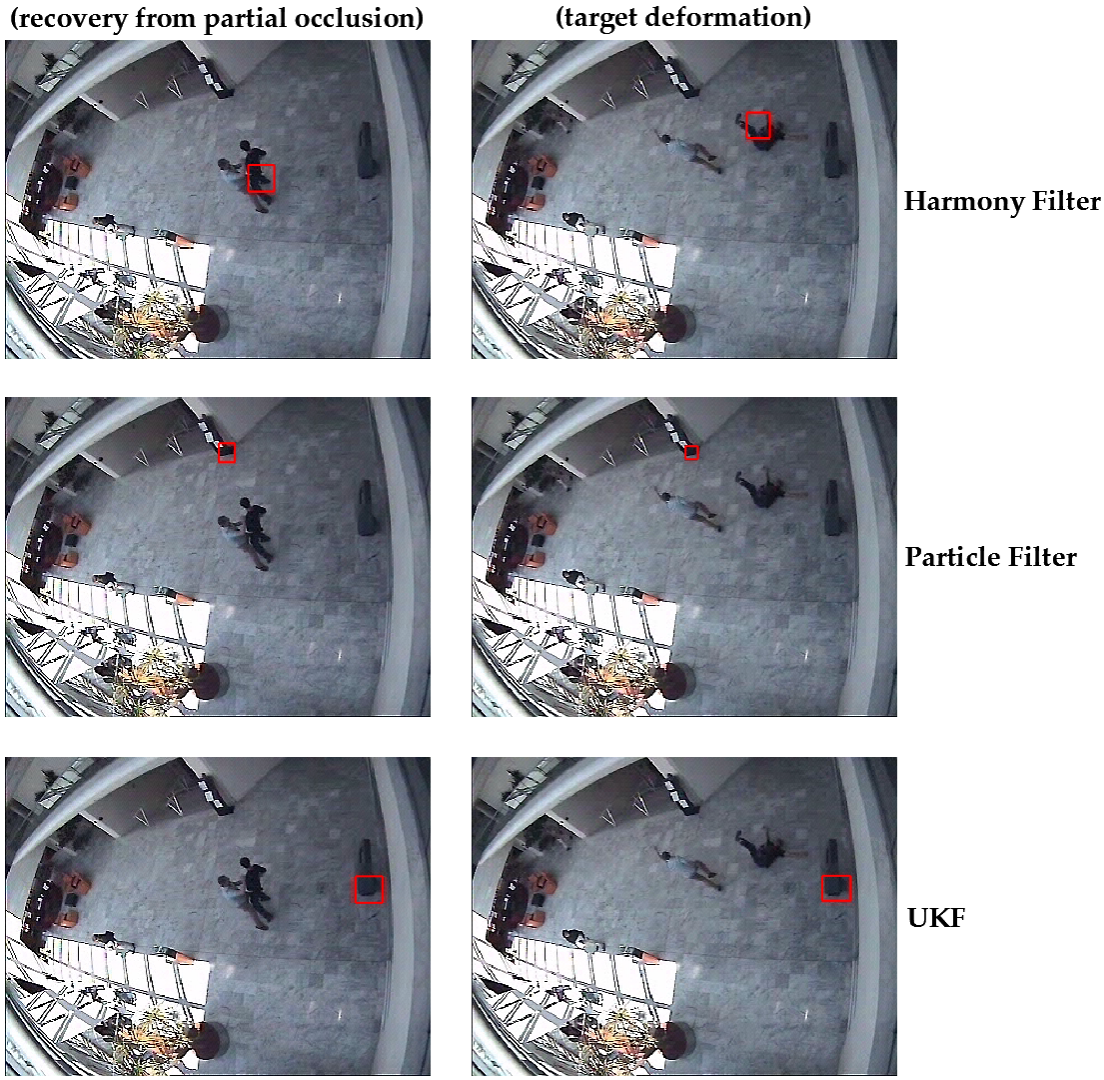

Figure 3.24: Two challenging tracking scenarios from the third example are presented. In the first scenario the trackers must recover from partial occlusion caused by one man moving in front of the other. In the second scenario the tracked target is significantly deformed when the man is knocked to the ground. In both scenarios only the harmony filter managed to robustly track the target.

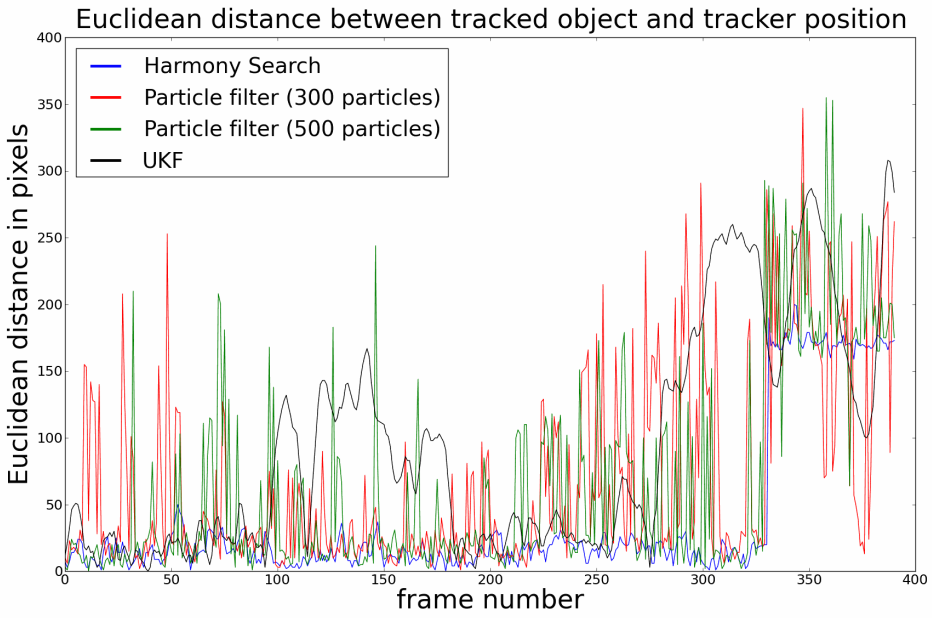

Figure 3.25: This graph compares the accuracy of various tracker implementations in the fighting men test sequence. The results are similar to those seen in the previous two examples and the harmony filter again gives the most accurate results on average.

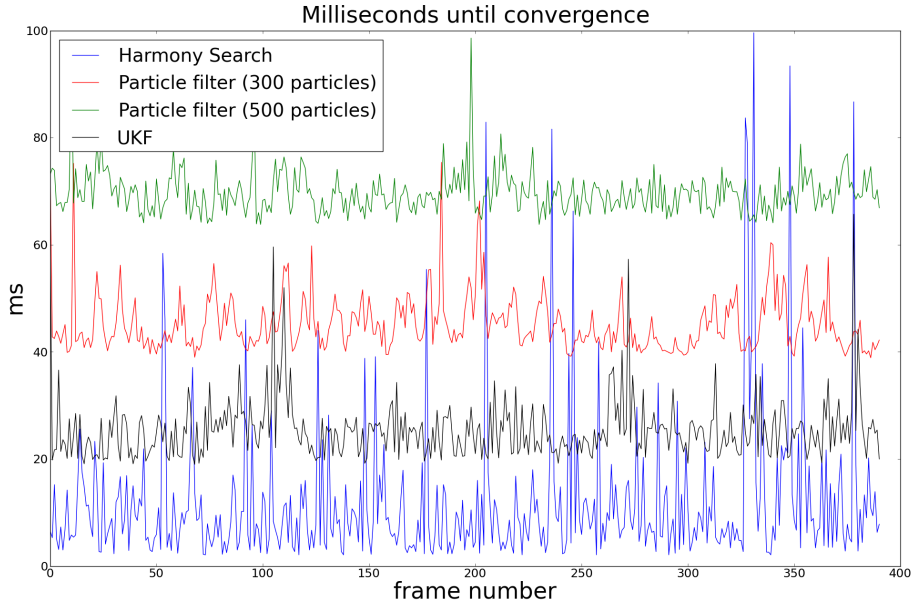

Figure 3.26: This graph compares the accuracy of various tracker implementations in the fighting men test sequence. Overall, the harmony filter converged quicker than any of the other algorithms even when the target was completely occluded and could not be located.

3.5 Discussion

From the results of the previous section we showed that the harmony filter performs very well under challenging situations and is robust to occlusion, rapid and erratic target movement and poor target modelling. Results from three independent experiments showed that the harmony filter can be twice as fast as the UKF and 4 times as fast as a particle filter implementation (using 300 particles). The harmony filter was also able to recover from extended periods of full or partial occlusion even when the UKF and particle filter implementations lost the target without ever recovering it again.

Even though the harmony filter was shown to be robust and efficient there are some problems and limitations that should be noted. One problem is that its speed is dependent on the tracking environment and thus cannot guarantee a constant frame rate. As seen in the speed comparisons of Figures 3.16, 3.21 and 3.26, the harmony filter is not consistently fast over all frames. While the speed of the particle filter and UKF remains consistent, we see spikes in the harmony filter's trace that indicate slower than usual convergence.

If one compares the location of these spikes with the corresponding locations in Figures 3.17, 3.20 and 3.25, one finds that slow convergence is usually accompanied by bad accuracy from all implementations. As previously illustrated, the frames with poor accuracy are most often the result of occlusion. During occlusion the harmony filter cannot find a good solution in the frame and local distractors that resemble the target can cause it to take a long time before the search is abandoned. The same situation can also occur during erratic motion when the target is usually blurred or deformed, or when the target is rotated as in the third test sequence (see Figure 3.23).

Constant performance is especially important in real-time applications since the available time until convergence is bounded. Usually the harmony filter returned the estimated location of the target before the other algorithms but in some cases it took the longest.

By limiting the maximum number of allowable iterations that Harmony Search can use, the maximum time until convergence can be capped and thus guarantee that the frame rate will never drop below a certain level. The frame rate still will not be constant since early convergence can still occur but it will be bounded above. However, be aware that capping the number of generations too low runs the risk of producing inaccurate

estimations. As long as the filter's speed is bounded to be no slower than a chosen limit, a non-constant speed is acceptable even in real-time applications.

One should also note that the same target model, colour histograms of rectangular patches, was used throughout all the experiments. In order to reach the performance levels seen in previous sections the parameters of the Harmony Search algorithm had to be optimised. It is likely that when a different target model, one that might be more appropriate to the problem, is used the parameters will have to be optimised again. This is something that has not been thoroughly investigated yet and might lead to stability issues when the harmony filter is used as a general tracking solution.

3.6 Chapter Conclusions

Algorithms for efficient and accurate visual tracking have been the subject of much computer vision research over the years. The most successful of these have all been based on statistical filters or Monte Carlo methods, for example, the Kalman filter and the particle filter. However, these algorithms often suffer in accuracy when the target's motion cannot be accurately modelled or when the sensor or motion noise cannot be assumed to be Gaussian.

The harmony search algorithm is able to find an optimal solution vector in a large multi-dimensional space without making the usual assumptions on the system noise or the shape of the search area. This makes it a potentially good algorithm for finding the best estimate of the location of a poorly modelled target in an image containing many similar potential targets. To test this hypothesis I designed a visual tracking system based on the harmony search algorithm called the harmony filter. The harmony filter is able to successfully track a poorly modelled target in real time and tests showed superior speed and accuracy when compared to other popular algorithms.

However, recent developments from research in the harmony search algorithm produced several improved or adapted versions of the original algorithm [3, 12, 15, 17, 90] (see also Section 2.6). Not all adaptations would likely lead to improved performance in the harmony filter as some are aimed at specific problems and not a generic improvement, but many could. The investigation of these algorithms as possible improvements to the harmony filter could lead to further improvements in speed and accuracy and is the theme of ongoing research.

Another aspect which could benefit from further research is accurate convergence detection. The importance of detecting convergence accurately has been thoroughly discussed in previous sections and future research into finding a better strategy could improve the speed of the harmony filter.

The current version of the harmony filter outperformed both the particle filter implementation and the UKF implementation in all of my experiments. This is the first time that the harmony search algorithm has been adapted for use in a visual tracking system and my initial results showed it to be a superior alternative to the popular particle and Kalman filter approaches.

4

Visual Correspondence Matching

> For now we see through a glass, darkly; but then face to face ...
>
> – *Paul to the Corinthians (1 Corinthians 13:12)*

In many computer vision applications one has to compare features from different images. These features are designed to describe parts of the image using a vector of real values. Two features that are sufficiently similar, as measured by some similarity metric, are considered to correspond to each other and might indicate that the objects associated with those features match. When multiple features from one image are uniquely corresponded with features in another image a visual correspondence set between the two images are formed. However, some features from one image might be sufficiently similar to multiple features from another image leading to ambiguous correspondence sets. Due to the nature of image feature modelling and natural visual ambiguities, this situation is the norm instead of an exception. The aim is therefore to find the visual correspondence set that successfully pairs the maximum number of image features without introducing contradictions due to visual ambiguities. This is known as the visual correspondence problem.

The visual correspondence problem is a challenging problem for several reasons mostly relating to the way visual features are modelled. Visual features appear differently when viewed from multiple angles and distances leading to ambiguous matches. For the same reason different features may appear very similar under certain conditions.

Modelling schemes that attempt to be invariant to scale, view angle and scene illumination are rare and tend to be computationally expensive making them impractical for real-time applications.

When simpler, computationally inexpensive, modelling schemes are used feature mismatches become more common and need to be filtered out of the final solution. RANSAC and graph cut algorithms are often used for filtering out feature correspondences that do not match but were identified as matches due to inaccurate modelling [91, 92]. However both of these algorithms suffer from high computational cost when the number of true correspondence matches are only a small proportion of the candidate matches. Silpa-Anan and Hartley report that true matches between scenes are often as low as 10% of the total number of candiate matches in visual localisation systems [93].

In this chapter a new method called the directed correspondence search (DCS) algorithm is presented for finding an accurate set of feature correspondences between two images. The DCS algorithm is based on the Harmony Search algorithm and was inspired by the success of the harmony filter discussed in the previous chapter.

4.1 Problem Statement

Before giving a formal definition of the visual correspondence problem we should first differentiate between sparse and dense visual correspondence. Dense visual correspondence aims to find a correspondence set that includes most or all of the pixels in an image. This approach is common in stereo vision applications where the aim is to construct a depth map with a one-to-one correspondence with the reference image [94]. In other words the correspondence set makes no distinction between the importance of individual pixels in the image but ideally provides an explicit match for each pixel in the reference image.

However, in this chapter the focus is on sparse correspondence sets. Instead of considering all the pixels, a few special features in the image are chosen and a correspondence set between features of the reference image and features in the alternate image is sought. This approach is not as computationally expensive as building dense correspondence sets as the set is much smaller with fewer data points. However, only the pixel depth can now be calculated only for a few select points in the image instead

a full depth map. For this reason the image features are carefully chosen to be both representative of the objects of interest in the image and to be easy to recognise. What is meant by *easy to recognise* and *representative* is not easily defined and is the topic of Section 4.2.1.

A sparse correspondence set is often used in structure from motion applications and visual navigation applications. By knowing the depth of certain recognisable features one can calculate camera ego-motion between different images. When used in a temporal sequence of images one can use this to calculate the camera odometry. This then allows one to calculate the odometry of a mobile platform using visual information only which is a main reason for its extensive use in robotic navigation [93, 95].

Let us then consider two images, one which we define as the reference image (I_R), and the other as the alternate image (I_A). In both these images we extract a set of visual features, one from the reference image ($F_R = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_m\}$) and one from the alternate image ($F_A = \{\mathbf{f}'_1, \mathbf{f}'_2, \ldots, \mathbf{f}_n\}$)[1]. Notice that the visual features are not simply pixel coordinates but are multidimensional vectors of real values that ideally describe the feature in some unique way. Notice also that the number of features extracted from the reference image is not necessarily equal to those extracted from the alternate image.

The aim is then to find the correspondence set, $C_{RA} = \{\mathbf{f}'_1, \mathbf{f}'_2, \ldots, \mathbf{f}'_m\}$, that matches each feature in F_R with a feature in F_A. Even though C_{RA} has the same number of elements as F_R, not all of the features in F_R will necessarily be matched to a feature in F_A. This means that some of the elements in C_{RA} are invalid matches. The invalid matches are indicated by the ϕ symbol.

An example illustrating this notation is given in Figure 4.1. Two images are shown of a light bulb from two slightly different angles. Let the image on the left be the reference image and the one on the right, the alternate image. Feature extraction is done on both images resulting in 7 features from the reference image and 9 features from the alternate image. Notice that the features cannot be matched on a one-to-one basis and that only four of the features in the reference image matches with features from the alternate image. This means that 3 of the elements in the correspondence set will be labelled as ϕ. The other labels indicate which feature from F_A correspond with the feature from F_R at that index. In other words, $C_{RA} = \{\mathbf{f}'_1, \phi, \mathbf{f}'_3, \phi, \phi, \mathbf{f}'_6, \mathbf{f}'_8\}$ means

[1]Here $'$ is used to differentiate features from F_R from those of F_A.

that $\mathbf{f}_1$ from F_R corresponds with $\mathbf{f}'_1$ from F_A, $\mathbf{f}_2$ does not correspond with any feature in F_A, $\mathbf{f}_7$ corresponds with $\mathbf{f}'_8$, etc.

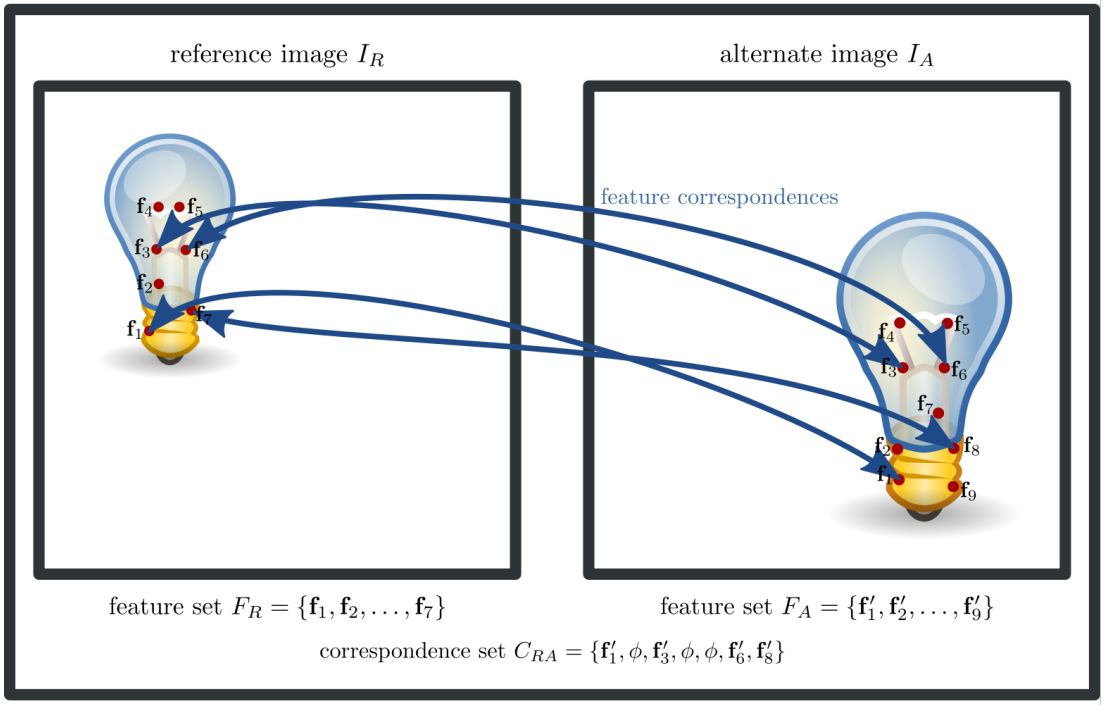

Figure 4.1: This diagram shows two slightly different views of a scene containing a light bulb. Image features, indicated by red dots, are extracted from both images. Four of the seven features in the reference image is successfully matched with features from the alternate image and the resulting correspondence set is shown.

Notice that no element in C_{RA} can ever be repeated since no feature in F_A may correspond to more than one feature in F_R. The only exception to this rule is when some features in F_R cannot be matched and the entry in C_{RA} becomes ϕ. Therefore, The only entry that may appear more than once in a correspondence set is ϕ. This uniqueness constraint is one of the aspects of this problem that make it challenging for heuristic optimisation methods like harmony search to solve. The HS improvisation step requires that each component can be independently changed to any value in the legal rage without having to take other component values into account. The uniqueness constraint prohibits the use of the original HS improvisation step as it could create invalid solutions that breaks this constraint.

This is however not the only challenging aspect of the problem. Objects in a scene generally do not have a consistent appearance when viewed from multiple angles. This can be true for even symmetric objects. For example, when light reflects off one side of an object and cast a shadow on the other side, the object's appearance can change dramatically depending on the viewpoint. This makes matching visual features very difficult as identical objects can therefore have very different visual features associated with them. Visual feature vectors are often designed to be invartiant to small changes in lighting or changes due to rotation but this is rarely sufficient to fully resolve the ambiguities. For this reason most correspondence matching algorithms assume that objects appear identical from all angles and rely on post-processing to filter out bad matches. Objects of this type have what is often called Lambertian surfaces [96]. This technically means that light reflects off its surface in such a way that the apparent brightness is identical from any viewpoint but the term is also more generally used to refer to surfaces that appear identical from all angles.

4.2 Overview of Current Approaches

In this section we investigate existing approaches to the visual correspondence problem. As we will see, most of the algorithms that are investigated in this section form part of a stereo vision system. This is not surprising as stereo vision is one of the most common applications for visual correspondence matching algorithms and this is where most of the recent development in correspondence matching is being done.

For the purpose of this review I will focus only on two main aspects that set visual correspondence matching algorithms apart. Firstly, the identification and creation of feature vectors that describe important parts of the image. Secondly, I investigate various ways that these feature vectors are matched and the process used to build the optimal correspondence set.

4.2.1 Feature Extraction and Feature Descriptors

Feature extraction is typically a two step process that first decides which areas of the image to select as descriptive features of the scene depicted, and then builds a feature descriptor from each area. These features should be chosen to be both repeatable and descriptive. In other words, a feature in one image should contain enough information

about the object it describes so that when that same object appears in another image, it can be recognised by its corresponding feature. If the feature is not repeatable then a feature associated with an object in one image will not necessarily lead to a similar feature being associated with that same object in another image. This means that correspondence of objects over multiple images will not be robust. Equally disruptive to the process is when a feature is not descriptive enough. Even when similar features are associated with the same object over multiple images, matching of these features become ambiguous in the presence of other objects and features when the features are not sufficiently descriptive.

In order to make features repeatable and robust they must be invariant to certain effects that images and objects within those images can undergo. A good feature descriptor aims to be invariant to as many of these effects as possible. However, it is rarely possible to be fully invariant to any of these effects especially when the feature's complexity is kept low due to limited computational resources.

Two of the most important characteristics that a good feature descriptor needs is rotation and scale invariance. This means that similar features should be associated with an object independent of the object's orientation and distance relative to the camera. This relates to my previous discussion on Lambertian surfaces and here it is usually also assumed that objects do not dramatically change their appearance depending on their orientation. Scale invariance ensures that objects that appear smaller due to their distance from the camera can be matched with identical objects that are closer to the camera and therefore appear larger.

Most of the corner detecting feature extraction techniques use feature descriptors that are rotation invariant. The most popular of these include the technique by Shi and Tomasi [97], the SIFT descriptor [98] and the SURF descriptor [99]. The SIFT and SURF descriptors are also scale invariant to an extent but tests showed that scale invariance can fail in certain conditions [100].

Illumination invariance is also important especially when images are illuminated by multiple light sources. Feature descriptors can usually be made illumination invariant by normalising all extracted features to have a specific mean intensity. This is the approach used to make SIFT and SURF descriptors illumination invariant but can be generically applied to most feature descriptors.

In an experiment to evaluate different feature descriptors, Botterill compared the efficiency of a selection of popular image descriptors [100]. He defined efficiency as the percentage of features in one image that correctly match with features in another, where a correct match is defined as a situation where a specific feature's most similar feature from another image corresponds to the same object. After comparing several techniques, three were chosen as the best candidates based on efficiency, robustness and speed. These were the SURF descriptor that was mentioned previously and a simple sum-of-square-difference patch descriptor centred on features using either the Harris [101] or FAST [102] feature extraction technique. The Harris features showed the highest efficiency and repeatability and was also almost as fast as the FAST features which is the fastest of the three. The SURF descriptor was the slowest but is also most invariant to scale and orientation changes.

4.2.2 Finding the Correspondence Set

Now that we have discussed feature extraction techniques and feature descriptors we can focus on matching these features into a correspondence set. A good correspondence set has two characteristics. First the features that correspond with each other must have high similarity as measured by the difference between the feature descriptors. Secondly, the correspondence set must represent a geometrically feasible camera movement between images.

For example, consider the two images in Figure 4.2. Both images show the same scene of a park but from two different camera positions. One of the trees and part of the dry fountain is visible in both images allowing us to match certain features from these areas between images. In this idealised example matches are made by paring off each feature from one image with the feature from the other image that is most similar to it. A likely result of this over simplistic approach is that many if not most of the matches will be incorrect. However in this example only two of the matches were incorrect. Correct matches, as indicated by red marker symbols, form a geometrically consistent set as shown by the yellow lines that connect matching pairs. However, two of the matches indicate a camera movement that is not consistent with the other matches. Even though their descriptors may be very similar, which would suggest a likely match, these matches should be discarded as they are not consistent with the camera movement indicated by the majority of the matches.

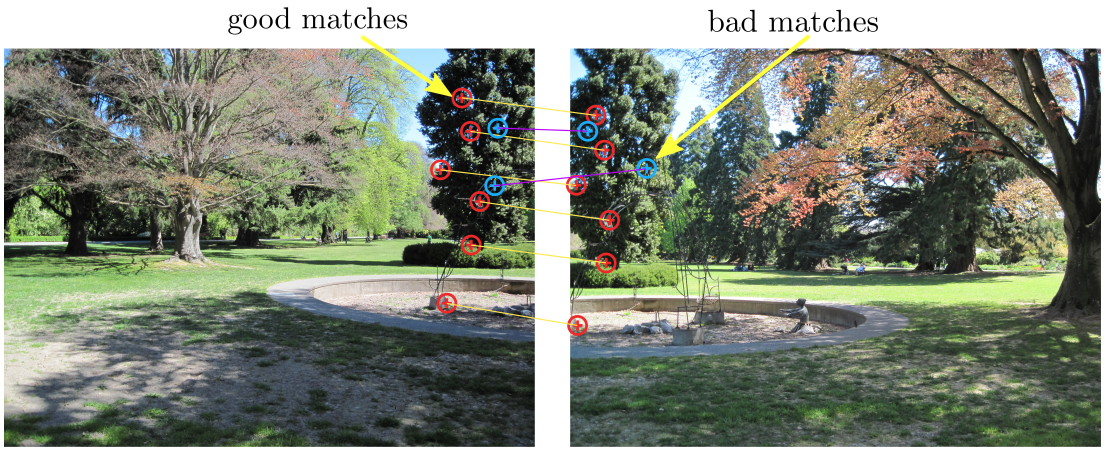

Figure 4.2: This diagram shows two images of a similar park scene. A tree and part of the dry fountain overlaps between images allowing us to match features between images. The good matches, indicated by red markers, all agree to a consistent camera movement as shown by the yellow lines. However, the bad matches, indicated by blue markers, suggest movement that is not consistent with the other matches.

An even more fundamental problem is how similarity between feature descriptors should be measured and which descriptors should be compared. There are many ways of comparing image descriptors but in this study my focus will be on descriptors that are efficiently compared using the sum-of-square-differences (SSD) method. With this method two descriptors, $\mathbf{f}_1$ and $\mathbf{f}_2$, are compared using the following equation:

$$\mathcal{C}(\mathbf{f}_1, \mathbf{f}_2) = \sum_{i=1}^{S} (f_1[i] - f_2[i])^2 \ , \tag{4.1}$$

where $\mathcal{C}$ is the SSD measure, $f_1[i]$ is the ith component of f_1 and S is the number of components in the descriptor.

Using the SSD measure, the feature that is most similar to $\mathbf{f}_1$ can be identified as the feature, $\mathbf{f}_2$, for which $\mathcal{C}(\mathbf{f}_1, \mathbf{f}_2)$ is a minimum. Let $F_R = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_m\}$ and $F_A = \{\mathbf{f}'_1, \mathbf{f}'_2, \ldots, \mathbf{f}'_n\}$ be two sets of features, one from the reference image and one from the alternate image. A naive approach to building the correspondence set C_{RA} would then be to calculate $\mathcal{C}(\mathbf{f}_i, \mathbf{f}'_j)$ for every combination of $\mathbf{f}_i$ and $\mathbf{f}'_j$ and set the values of C_{RA} to correspond to the features for which $\mathcal{C}(\mathbf{f}_i, \mathbf{f}'_j)$ was minimised.

However, there are several problems with this strategy. First, it ignores the geometric constraints imposed by consistent camera movement as illustrated in Figure 4.2.

Also realise that feature descriptors are not perfect representations of the features they are associated with. Therefore, two descriptors that are similar do not necessarily represent the same image feature. Matches that are based purely on finding the most similar descriptors will often fail for this reason. This problem is especially prevalent in the often encountered situation where some features are only detected in one image and have no correct correspondence in the alternate image. A system based solely on similarity scores will try to find a match even when none exist and will therefore necessarily introduce mismatches into the correspondence set.

One way to improve the accuracy of this method is to ensure that there are no similar matches possible when two descriptors are matched with each other. This improvement is implemented by Brown et al. and Nistér et al. by ensuring that the match similarities for the best and second best match candidates are not within 20% similarity threshold with each other [103, 104]. This somewhat arbitrary threshold helps to ensure that the most ambiguous matches are excluded from the correspondence set but the problem is not completely alleviated.

Not only does this approach lead to many mismatches in the correspondence set, but it is also computationally expensive. The SSD measure will have to be calculated $m \times n$ times to create the correspondence set which typically means that $\mathcal{C}$ needs to be calculated several thousand times for each correspondence set. A slightly more efficient approach is to use a balanced binary tree called a *kd*-tree to find the most similar descriptor using fewer evaluations of $\mathcal{C}$ [105]. Instead of adding features to an unsorted list, features are sorted into a balanced tree by thresholding on certain component values determined by the median of the components with the highest variance. Matching is then accomplished by traversing this tree towards a leaf node that is sufficiently similar to constitute a match. This method minimises evaluation of $\mathcal{C}$ by eliminating most of the evaluations that would likely not lead to a match.

4.2.2.1 RANSAC

However, the previously mentioned method alone still does not take into account geometric constraints. A more robust method is clearly needed. The Random Sample Consensus (RANSAC) method is the most popular method for solving this type of problem and has been used extensively in stereo vision applications [106–108]. RANSAC is able to robustly estimate a model's parameters from a set of observations even when

those observations are corrupted by noise and outliers. This can be applied to the correspondence problem by noticing that the geometric constraints imposed by consistent camera movement form a mathematical model that all the correct matches must agree with. The features and their initial matches based on similarity are the observations corrupted by noise and multiple outliers. From these observations RANSAC tries to extract the largest set of features that are consistent with a single camera motion model. This set is considered to be the model's parameters.

RANSAC assumes that most of the observational data consists of points that are inliers, those that are consistent with the model, and that some of the points are outliers, those that are not consistent with the model. It attempts to find the set of inliers by iteratively testing a random sample set. This set is considered as the set of hypothetical inliers and the model that is built from it is considered as the best inlier model for all the observations. To test this hypothetical model all other data points, those not in the inliers set, are fitted to the model and those points that fit well are included into the consensus set. If the consensus set is larger than the current best set, the model is considered good and the current consensus set becomes the new best inlier set. RANSAC then continues to sample random sets of points in an attempt to find a better model. This continues until the maximum number of iterations is reached. Pseudo code for the RANSAC algorithm is given in Algorithm 4.1.

RANSAC has been used for many years in computer vision applications and several improvements has since been suggested. The MLESAC and MLS algorithms are based on RANSAC but modify the model selection step so that, for example, the model with the highest estimated likelihood is selected instead of the one with the highest inlier count [109, 110]. This can increase the quality of the model leading to smaller fitting errors in the final consensus set.

In order to improve the speed of RANSAC many researchers have attempted to modify the hypothesis set selection strategy. By improving the selection strategy one can reach the best consensus set sooner using less iterations. The large number of iterations that are typically required to find the best consensus set is the main reason for the often slow speed of RANSAC.

To improve the selection strategy, probabilities are assigned to the observations based on their likelihood of being part of the inlier set. Instead of randomly sampling points from the set of observations, we now sample points which are more likely to be

in the inlier set with a higher probability than those that are not. It is often simple to assign probabilities to feature matches based on their match qualities with the best matches getting the highest probabilities. One can also assign lower probabilities to matches from features that are similar to multiple features and are thus less likely to be correct. This improves the quality of hypothetical inlier sets and lead to quicker discovery of the best consensus set. Some examples of algorithms that use this strategy to improve RANSAC in include BAYSAC and PROSAC [100, 111].

4.2.2.2 Graph Cut Methods

Graph cut methods are inspired by the discovery that energy minimisation (or the optimisation of some objective function) can be efficiently performed by modelling the problem as a set of edges and vertices arranged in a graph. Typically the pixels in the image is represented by vertices in the graph with neighbouring pixels connected by edges called neighbour links. Special terminal vertices, which represent pixel labels, are also added to the graph and these are also connected to the pixel vertices with special edges called terminal links.

Each edge has a weight associated with it that is set by the energy function. When the problem is correctly modelled as a graph, combinatorial optimisation theory teaches that the optimal labelling, that is the one that minimises the energy function, is represented by the *minimal graph cut* [112]. The minimal graph cut is a way to partition the graph such that the sum of the edge weights associated with the edges that were cut in the partition is minimised [113]. An example of a pixel labelling problem modelled as a graph is shown in Figure 4.3.

Graph cut minimisation has been used successfully in image segmentation and clustering applications and has shown to be more efficient than most state-of-the-art methods in these applications [114, 115]. The visual correspondence problem is also essentially a pixel labelling problem that can be solved through energy minimisation.

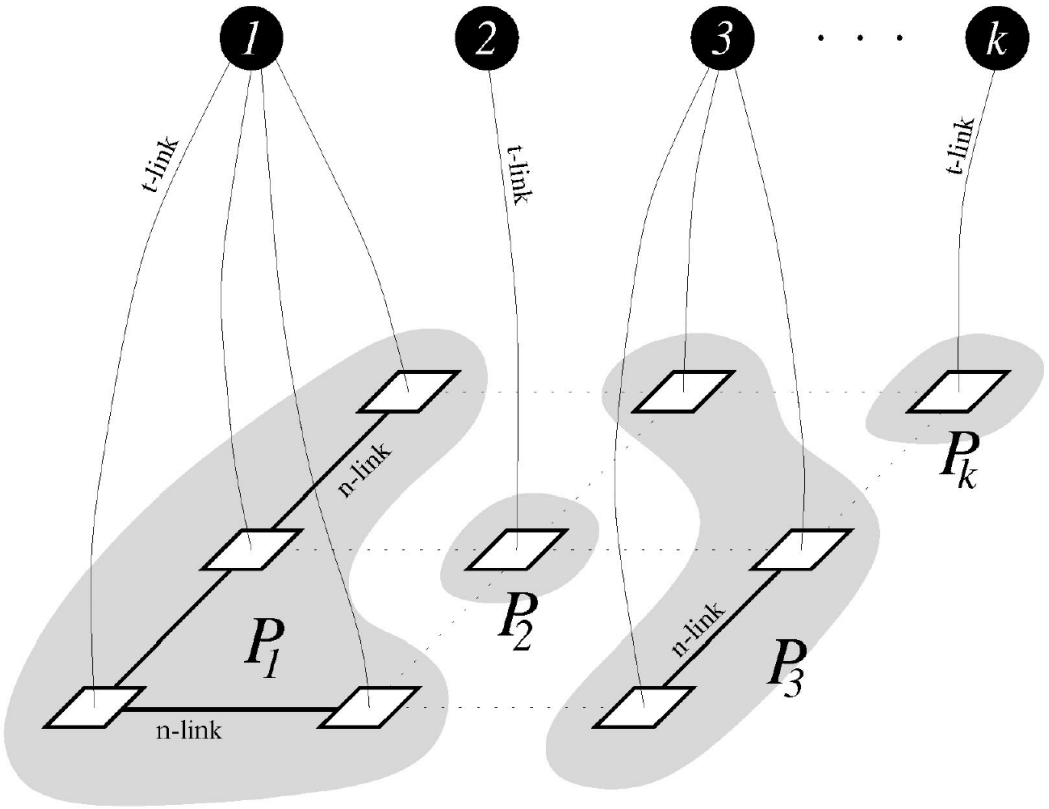

Figure 4.3: In this diagram an image labelling problem is represented as a graph. The square vertices are pixels in the image and the circular vertices are image labels. All the image labels are connected to all the pixel vertices with terminal links (t-link) (some links are omitted for clarity) and pixels are connected to their neighbours with neighbour links (n-link). The partitions p_k to P_k results from a single graph cut that associates pixels with labels and represent a specific pixel labelling. This image was taken from [112].

input : C_{RA} = the set of matched features (observations)
input : M = a camera motion model that can be fitted to matches
input : n = the minimum size of an inlier set
input : $G_{\max}$ = the total number of iterations
input : t = threshold value to decide when a match fits the model
output: M_{best} = the overall best model
output: S_{best} = the best inlier set
output: e_{best} = the fitting error from the best model

$i = 0$
$M_{\text{best}} = \{\}$
$S_{\text{best}} = \{\}$
$e_{\text{best}} = \infty$
while $i < G_{\max}$ **do**
  $S_{\text{inliers}} = n$ random matches
  M_{inliers} = the model fitted to S_{inliers}
  $S_{\text{consensus}} = S_{\text{inliers}}$
  foreach match $\in C_{RA}$ and $\notin S_{\text{inliers}}$ **do**
    if match fits M_{inliers} with error $< t$ **then**
      add match to $S_{\text{consensus}}$
    end
  end
  if $|S_{\text{consensus}}| > |S_{\text{best}}|$ **then** `// this is a good model`
    $S_{\text{best}} = S_{\text{consensus}}$
  end
  $i = i + 1$
end
M_{best} = model fitted to all matches in S_{best}
e_{best} = fitting error from M_{best}
RETURN $M_{\text{best}},\ S_{\text{best}},\ e_{\text{best}}$

Algorithm 4.1: The RANSAC algorithm

In traditional energy minimisation the aim is to minimise a energy function that has the following form:

$$E(C_{RA}) = K \cdot E_{\text{smooth}}(C_{RA}) + E_{\text{data}}(C_{RA}), \tag{4.2}$$

where C_{RA} is a correspondence set between the features of the reference image and the alternate image (see Figure 4.1). The first term is a smoothness term that imposes a penalty on solutions that violate spatial smoothness while the second term penalises solution that are inconsistent with the observed data. K is a weighting constant that determines the relative importance between the two terms. When searching for a correspondence set, the data term is typically built around a feature matching measure and the SSD measure introduced in Equation (4.1) is often used. The smoothness term is more difficult to choose and many variations have been proposed [112].

A popular smoothness term that is often used in graph cut minimisation is the Potts model. The Potts model preserves discontinuities in the image and has various other features that make it preferable for use in graph cut minimisation [112]. The Potts model for the smoothness term is defined as

$$E_{\text{smooth}}(C_{RA}) = \sum_{p,q \in \mathcal{N}} T(f_p \neq f_q) \ , \tag{4.3}$$

where $T(\cdot)$ is 1 if its argument is true, and 0 otherwise. The term is only calculated for those pixels in $\mathcal{N}$ which is the set of interacting pixels and is typically those pixels that are adjacent.

The Potts model encourages solutions consisting of regions where pixels in the same region have equal labels. This type of term is sometimes informally called a piecewise constant model [112]. The Potts model is especially useful when the labels are unordered and the number of labels is small. It has been shown that the Potts model based energy minimisation problem can be directly reduced to the multiway graph cut problem [112]. It can therefore be proven that the global minimum for a Potts model based energy function can be computed by finding the minimum cost graph cut on an appropriately constructed graph.

Therefore, one can solve the visual correspondence problem by first modelling it as an appropriately constructed graph and then using known combinatorial optimisation

techniques to find the minimum cost multiway graph cut for that graph. The edges between the terminal vertices (representing the labels) and the pixel vertices then define the optimal correspondence set C_{RA}. Efficient algorithms for the calculation of such correspondence sets using this technique has been suggested by several researchers and some of the best stereo depth maps have been constructed in this way [92, 112, 116, 117].

One of the main advantages of graph methods is that many of them are guaranteed to converge to a local optima within a bounded number of iterations. Boykov et al. showed that in practice most of the optimisation is done in the first iteration and termination is usually within the first few iterations [112]. This fast convergence combined with a bounded execution time makes graph cut methods a popular alternative to RANSAC based approaches.

4.3 The DCS Algorithm

In the previous section we saw that the visual correspondence problem can be modelled as an energy minimisation problem. While the energy function is typically non-convex and difficult to minimise using conventional means, it can be considered as an objective function and therefore minimised using optimisation algorithms discussed in previous sections. Therefore, it should be possible to use harmony search to find a solution. To do this we need to model potential solutions as vectors of decision variables that are arranged appropriately for the improvisation process of harmony search. We also need an appropriate objective function similar to those seen in the previous section to score and compare solution vectors.

As we will see in the sections that follow, this type of problem is more difficult than it seems to model appropriately for optimisation using harmony search. Solutions to this problem have unique constraints and the search space lacks features that are traditionally used to accelerate convergence like wide basins of attraction. In this section Directed Correspondence Search (DCS) is introduced as an adaptation of harmony search to solve the visual correspondence problem.

4.3.1 Modelling the Decision Variables

In Figure 4.1 we introduced a method of representing correspondence sets as a vector with length equal to the number of features in the reference image. Each component of

this vector indicates which feature in the alternate image corresponds with the feature in the reference image located at that index. The ϕ symbol was introduced to indicate features in the reference image that could not be matched with any features in the alternate image.

Notice that there may be more features in the reference image than in the alternate image. Since features have to be uniquely matched and cannot be repeated in the correspondence set some components will be set to ϕ in this case, even when all the features in the alternate image was matched.

When used as a solution vector in the harmony memory, the decision variables are therefore feature labels from the alternate image. However, it is important to remember that when the solution vectors are modelled in this way, the decision variables are just labels and should not be treated as numerical values. This has a major effect on the improvisation process that rely on the fact that decision variables are values that can be incremented and decremented to find neighbouring solutions in the search space.

Firstly, the fact that values in the solution vector have to be unique and cannot be repeated not only constrains the size of the search space, but also makes many candidate solutions that would be created during improvisation, invalid. The improvisation process assumes that each decision variable in the solution vector can be optimised independently and that any value between the upper and lower bound leads to a valid solution vector. This is clearly not the case in correspondence sets.

Let C_{RA} be a correspondence set constructed from the features in F_R and F_A (see Figure 4.1). In a correspondence set, C_{RA}, each decision variable has to be chosen from the set of values found in F_A that have not been used in any other component of C_{RA}. In this way each time a component value is improvised every other component in the solution vector has to be taken into account to make sure a valid improvisation is reached. Clearly the standard improvisation procedure will have to be adapted to maintain valid solutions in the harmony memory.

As previously mentioned, another assumption that many optimisation algorithms, including harmony search, make is that good solutions are found in clusters in the search space. In other words, when one good solution is found it is expected that other good solutions will be found in the same vicinity. This assumption is valid in problems where good solutions lie in a basin of attraction but cannot always be made when searching for optimal correspondence sets.

To illustrate how this assumption affects improvisation consider the following accurate correspondence set $C_{RA} = \{f'_4, f'_1, \phi, f'_2, \phi\}$. This is interpreted as f_1 corresponding to f'_4, f_2 corresponding to f'_1 and f_4 corresponding to f'_2. Another set $\hat{C}_{RA} = \{f'_5, f'_2, \phi, f'_1, \phi\}$ is adjacent to C_{RA} in the search space but represents a completely different solution that could have a much lower fitness than C_{RA}. $\tilde{C}_{RA} = \{f'_4, f'_2, \phi, f'_2, \phi\}$ is also adjacent to C_{RA} but does not even represent a valid solution since f'_2 corresponds with more than one feature. Slightly adjusting components of C_{RA} through pitch adjustment in the hope of finding other good solutions in the same vicinity therefore simply degrades to inefficient random search. The root cause of this problem is that the numbers in the solution set are arbitrary labels and not values that have special relationships with each other.

If harmony search is to be adapted for solving the correspondence problem two changes will have to be made to the improvisation process. First component values cannot be optimised independently from one another as each value has to be unique. Second, the pitch adjustment operation needs to be replaced with an operation that adjusts C_{RA} in such a way that the new candidate solution remains valid and is likely to have a similar (ideally better) fitness weight.

4.3.2 Directed Search Strategy

The main idea behind DCS is to replace the pitch adjustment operation in harmony search with a swap operation that is guided by the objective function. The solution vectors are modelled as explained in the previous section and an adapted harmony search optimiser is used to find the optimal solution vector. Since the pitch adjustment operator is removed the PAR and FW parameters are also removed from DCS as they are no longer needed. An overview of DCS is illustrated in Figure 4.4.

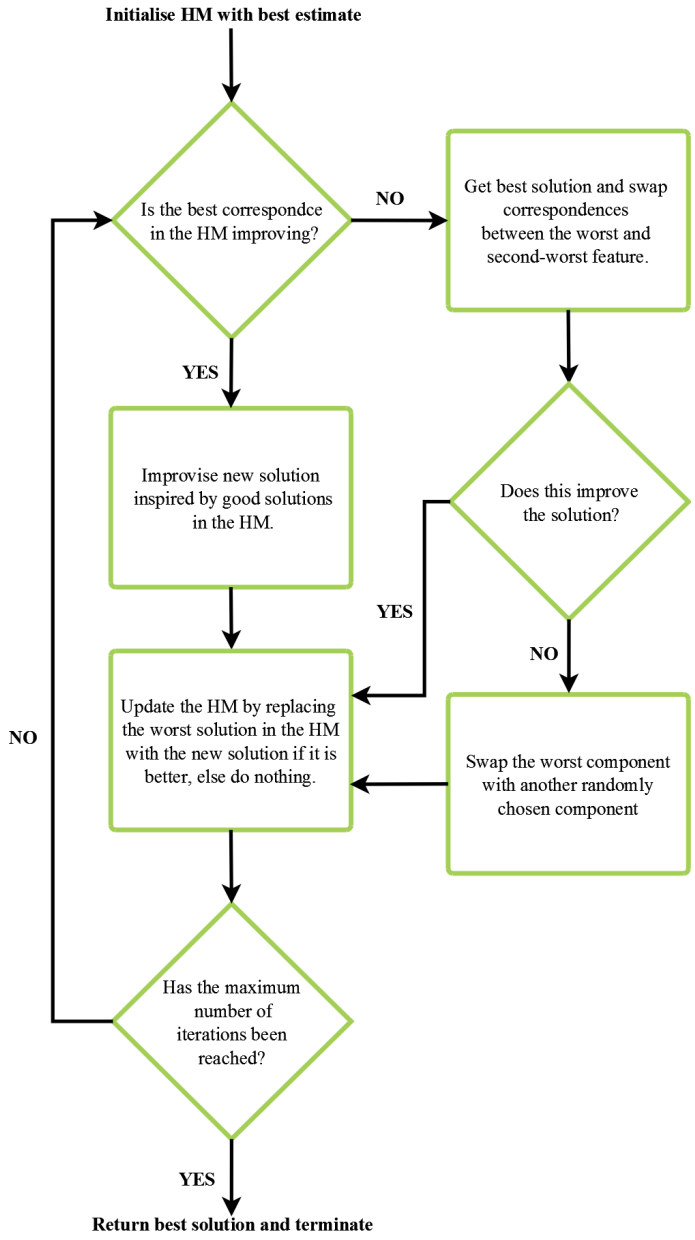

Figure 4.4: This diagram shows the main steps of the DCS algorithm. The main difference between it and standard harmony search lies in the addition of the swap operation that injects diversity into the HM without violating the uniqueness constraint or generating invalid solutions. Notice also that in the improvisation step only memory consideration and random selection are used. Pitch adjustment is never used to generate improvisations in DCS.

The main loop from DCS is very similar to harmony search and the swap operation is only used when no further progress is made through improvisation using only memory consideration and random selection. The swap operation should not performed frequently as this can lead to a premature loss in diversity in the HM and cause the optimiser to get stuck in a local optima. The swap operation is only performed when the number of consecutive idle operations reaches a constant maximum. If the swap operation is successful in generating a solution that updates the HM, the number of idle iterations is reset to zero and execution is returned to the main loop without the swap operation.

4.3.3 The Uniqueness Constraint

In the Harmony Search algorithm a new candidate solution is built up component-by-component during improvisation. To ensure that each component of the new candidate has a unique value or possibly ϕ, a set of possible values for each component is created before improvisation. The set, referred to as $\mathbf{V}$, includes one copy of each possible feature correspondence and ϕ. Each time a new component value is improvised it is checked to ensure that the value is present in $\mathbf{V}$. If it is not then that value has already been used and cannot be used again which leads to the generation of another value until a valid one is found. When a valid value is found it is accepted and removed from $\mathbf{V}$ to prevent its further use. The only exception is when ϕ is used as the new value. In this case the value is accepted without consulting or updating $\mathbf{V}$ since ϕ can be used multiple times. Pseudo code for the improvisation step of the DCS algorithm is given in Algorithm .4.

This step is the primary means of improvising new solutions and maintaining diversity in the HM. The only time that improvisations can come from a different source is when the maximum number of idle iterations have been reached, in which case the swap operation is used.

A similar process is used to generate candidate solutions during initialisation of the HM. When prior knowledge is available at least one of the entries in the HM will be initialised to be equal to the prior. Experimental results show that adding two copies of the prior to the HM gives best results when a small HM (HMS $\approx$ 7) is used. The remaining entries are generated with random values chosen to adhere to the uniqueness constraint.

4.3.4 The Swap Operation

Even when the uniqueness constraint is taken into account, the standard Harmony Search algorithm often fails to find the correct correspondence set in a reasonable number of iterations. Random selection and memory consideration were designed to work together to thoroughly explore the search space while simultaneously converging to the optimum using the HM. However, after a few iterations the uniqueness constraint combined with the lack of pitch adjustment makes memory consideration ineffective to converge to the optimum.

Once most of the correct correspondences are present in the HM further optimisation can only be made when two values from a candidate in the HM are swapped. This action is impossible using memory consideration since a swap operation changes two values simultaneously while memory consideration updates one component at a time. This would not usually matter since a swap operation can be done in two consecutive steps one component at a time. However, the first step makes both values to be swapped equal which breaks the uniqueness constraint. Any other series of steps that can be used to swap two component values using random selection to provide the necessary entries in the HM, creates intermediate candidates with fitness values that are too low to be introduced to the HM. Therefore only a single operation that changes both values simultaneously can create an improved candidate and further the convergence process.

As previously mentioned, swap operations are triggered when the algorithm detects that normal operation is not furthering convergence satisfactorily. It is detected by counting the number of idle iterations or iterations in which the HM was not updated. When the number of consecutive idle iterations exceeds a given threshold a swap operation is triggered.

The swap operation is always performed on the best candidate in the HM. In this way it is more likely that a new candidate will be created that improves on the fitness of the worst candidate in the HM. Other candidates in the HM can also be chosen and a more complicated selection scheme can used but experimentation showed that simply choosing the best candidate in HM produces the best results.

It is highly unlikely that swapping two randomly chosen values would create a new candidate with a high fitness score. We rather choose to swap the values of the components that contribute most to decreasing the fitness score of the candidate. In

this way the search process is guided towards an area where it is most likely that better solutions will be found. We therefore choose the worst and the second worst components and swap them to form a new candidate solution. The worst component in a solution vector is determined by the objective function which will be discussed in the next section. If the initial swap operation fails to improve the HM a random swap is performed instead by swapping the worst component with a randomly chosen one.

4.3.5 The Objective Function

The design of the objective function is crucial to the DCS algorithm. Not only does it determine which candidates are kept in HM and ultimately determine the optimum, but it also guides the swap operation by determining which components should be swapped. Its design is significantly influenced by the way image features are compared and the expected accuracy with which positive matches can be identified using only these comparisons. The function is designed to give more or less weight to feature comparisons depending on the method used.

A good correspondence set not only contains strong feature matches but must also be geometrically consistent. To measure this we make the static world assumption that states that all observed features are static and all perceived feature movement between images are due to camera movement only. Since only the camera is moving all perceived feature movement must be in the same direction. When there is a lot of diversity in the perceived feature movement between images in a correspondence set, it means that either the scene was corrupted with real moving objects or the correspondence set is incorrect.

Feature movement is calculated by using the 3D coordinates of the features in the images. These are calculated independently using other sensors like laser scanners or by using multiple view geometry [108]. The local 3D coordinates of each feature in one image is subtracted from the 3D coordinates of the corresponding feature in the other image. This gives the movement vector for that feature given the correspondence set that is being tested. An illustration of this process is given in Figure 4.5.

The mean and variance of the motion vectors over all features are then calculated. The mean is used to determine which feature correspondences contribute most to lowering the fitness of the correspondence set. The movement that deviates most from the

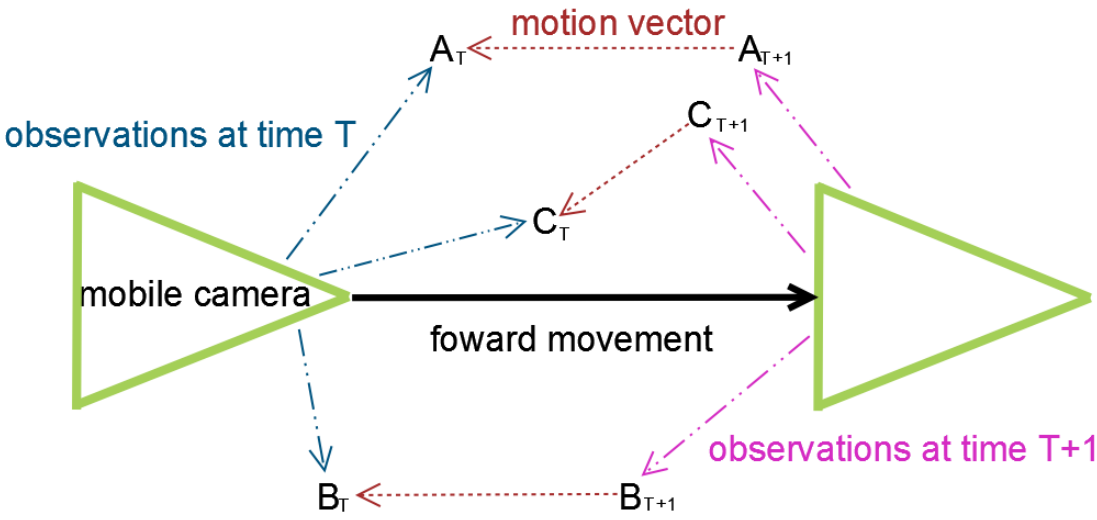

Figure 4.5: Perceived feature movement due to camera movement is measured using motion vectors. At time T the camera observes 3 features. Then it moves forward and observes the same features. From this position it appears that 2 of the features moved backwards while one moved diagonally down. The most likely interpretation is that two of the correspondences are correct while the one that moved differently from the others is incorrect.

mean corresponds to the worst feature correspondence and represents the component that would likely benefit most from a swap operation.

Notice however, that due to perspective geometry, objects that are further away from the camera appear to move less than objects that are close by. To cancel this effect the motion vectors are normalised before being used in the weight calculation. Once normalised the mean and variance over all features are calculated.

The variance is used to calculate the fitness weight. A small variance indicates a low diversity in movement vectors and leads to a high fitness. A large variance indicates inconsistent feature movement and leads to a low fitness. The variance term is added to a feature match quality term that is dependent on the choice of feature representation. In this case it is simply the SSD measure between corresponding features which was introduced in Section 4.2.2.

In order to discourage DCS from generating correspondence sets with too many ϕ values, a starvation penalty term is added to the weight to make sure that a sufficient number of good correspondences are found. The penalty is designed so that the fitness is severely penalised below a certain threshold but makes a small contribution if plenty

of correspondences are found. The threshold is dependent on the application for which the correspondence set is used for. For example, many good correspondences are needed for accurate visual navigation and the fitness should be severely lowered if there are less than the required minimum number of correspondences.

The equation that combines these terms and calculates the fitness is as follows.

$$f(C_{RA}) = -(\sum_{i=1}^{m} \mathcal{C}(C_{RA}[i], F_R[i]) + \alpha\sigma + \beta e^{-c\tau}) \ , \tag{4.4}$$

where $\sum_{i=1}^{m} \mathcal{C}(C_{RA}[i], F_R[i])$ is the SSD measure over all feature representations. The two weights, α and β, determine the relative importance of the three terms in the final score while σ is the motion vector variance. The final term is the starvation penalty modelled as an exponential function with τ controlling the shape and c the number of correspondences in the set.

4.3.6 Initialisation

Before optimisation using DCS starts the HM is first initialised with two copies of the prior correspondence set estimate. The remaining vectors are generated randomly. The generation of the prior estimate is very important to the efficiency of the optimiser as the prior usually becomes the initial best solution and therefore will have a large influence on the direction of the search. This step should also be as fast as possible to allow the main loop of the optimiser the maximum number of iterations when computational resources are limited.

In the DCS algorithm image features are modelled using the SURF descriptors [99] mentioned in Section 4.2.1. In order to quickly build an initial estimate for the correspondence set, a fast but naive approach is used to match SURF descriptors. Each feature in the reference image is considered independently and compared with all the remaining unmatched features in the alternate image using SSD. The match with the lowest SSD measure is then considered as the matching correspondence if the SSD is sufficiently low that a true match is likely. If none of the SSD measures are sufficiently low the feature is considered unmatched and ϕ is inserted into the prior set at that index. This process is continued until all the indices in the prior set has been set.

This process generates a very rough estimate of the correct correspondence set as geometric consistency is completely ignored. Errors are also introduced by the

elimination of possibly correct matches due to them being incorrectly matched to a previous feature in the reference image. In other words the first few features in the reference image have more options to match than those that are processed later. This puts an incorrect bias on some features and leads to incorrect matches for many of the features that are processed last.

Even though the prior is far from perfect it is good enough to initialise the HM and gives the search process a good stating point. This method of initialisation is also fast since the list of features in the reference image is only iterated once.

4.3.7 Algorithm

As previously explained, the main loop of DCS is the harmony search improvise-and-update step but without any pitch adjustment. This means that there is no need for the PAR or FW parameters. However, DCS introduces a new parameter that controls the frequency of the swap operation called the swap rate (SR). The SR is defined as the maximum number of idle iterations that can be performed before DCS switches from the normal main loop to the swap operation. Full pseudo code for the DCS algorithm is given in Algorithm .5 to .7.

Notice also that no convergence detection is done and that optimisation only ends when the total number of iterations have been reached. I found that, due to the introduction of the swap operation, traditional convergence tests fail to accurately detect convergence and that their addition does not speed up the algorithm. One could test for convergence by monitoring the motion covariance (see Algorithm .7 line 15) and this has been experimented with. However, this also did not significantly improve the speed of the algorithm and sometimes caused premature convergence when most of the features are unmatched.

Another convergence test that was considered was idle iteration testing. This was especially convenient since the number of idle iterations are already monitored to control the swap operation. However, this approach was found to be even more prone to converge prematurely. In the end the best results came from simply terminating after a maximum number of iterations which also keeps an upper bound on the running time.

4.4 Results

In this section I evaluate the DCS algorithm using an example captured using the Bumblebee 2 camera from Point Grey Research[1]. I use a stereo camera to simplify the calculation of 3D feature coordinates but do not use the DCS algorithm for calculating stereo correspondences between the sensors of the Bumblebee 2. A simpler algorithm specific to this type of stereo camera is used for this. In this example DCS was used to find correspondences in real time between new features from the current image from the camera, and features stored in a digital map. The application was a visual odometry system that uses multiple view odometry to calculate 3D positions from SURF image features captured from a mobile calibrated stereo camera. Some of these features are then stored in a digital map that represents knowledge of the camera's surroundings. By matching new features from the mobile camera with those in the map using DCS, the camera ego-motion can be calculated.

The example that is considered is shown in Figure 4.6. In the first image 112 image features are extracted and saved in the digital map. The camera is then moved forward one meter and another image is extracted. From this second image 128 features are extracted. DCS is then used to build a correspondence set between the features from the first image and those of the second.

DCS is initialised with a prior correspondence set generated by matching features using only the feature descriptors and choosing randomly where this method is ambiguous as explained in previous sections. The prior solution identified 39 correspondences from the two images but many of these were not accurate and the motion vector variance over all correspondences was large for this set indicating inconsistent geometry.

Once initialised with this prior the DCS algorithm optimised the correspondence set in 600 iterations producing a much more accurate set of 38 correspondences. The optimised set has a 3D motion vector variance of [0.125325, 0.0113381, 0.281385]. Compared to the prior's variance of [2.51285, 0.133806, 8.65715] the optimised set's variance is significantly smaller.

A trace of the optimisation process illustrating the evolution of the best and worst solution in the HM is shown in Figure 4.7. Only the first 200 iterations of a total of 600 is shown as this is where most of the optimisation occurs. Notice that after an initial few

[1] `http://www.ptgrey.com` (February 2011)

Figure 4.6: Two images of the same scene taken from different positions. The top image is taken first. Then the camera is moved one meter forward where the second image is taken.

iterations most of the progress is made through swap operations since normal memory consideration fails to make further improvements. Notice also the rapid progress made during the first 50 iterations. After 50 iterations the quality of the best solution in the HM is already more than 15 times better than the prior it was initialised with.

By carefully investigating the position of the crosses in the graph (that represent swap operations), and comparing this with the positions where progress is made to a better correspondence set, we find a pattern forming. This pattern shows several memory consideration steps followed by one or two swap operations that often result in an improved solution. It is very interesting to notice that, in the DCS improvisation

process, memory consideration is mainly responsible for maintaining diversity in the initial search for improved solutions and usually does not play a large role in the later stages of optimisation. However, if we were to argue that memory consideration is not necessary after the first few iterations and that further optimisation should only be done using swap operations, all the solutions in the HM would quickly degrade to simply being slight variations of the best solution considerably slowing down convergence. DCS uses memory consideration to maintain diversity in the HM without resorting to randomly perturbing solutions in the hope that the optimiser will stumble across better solutions.

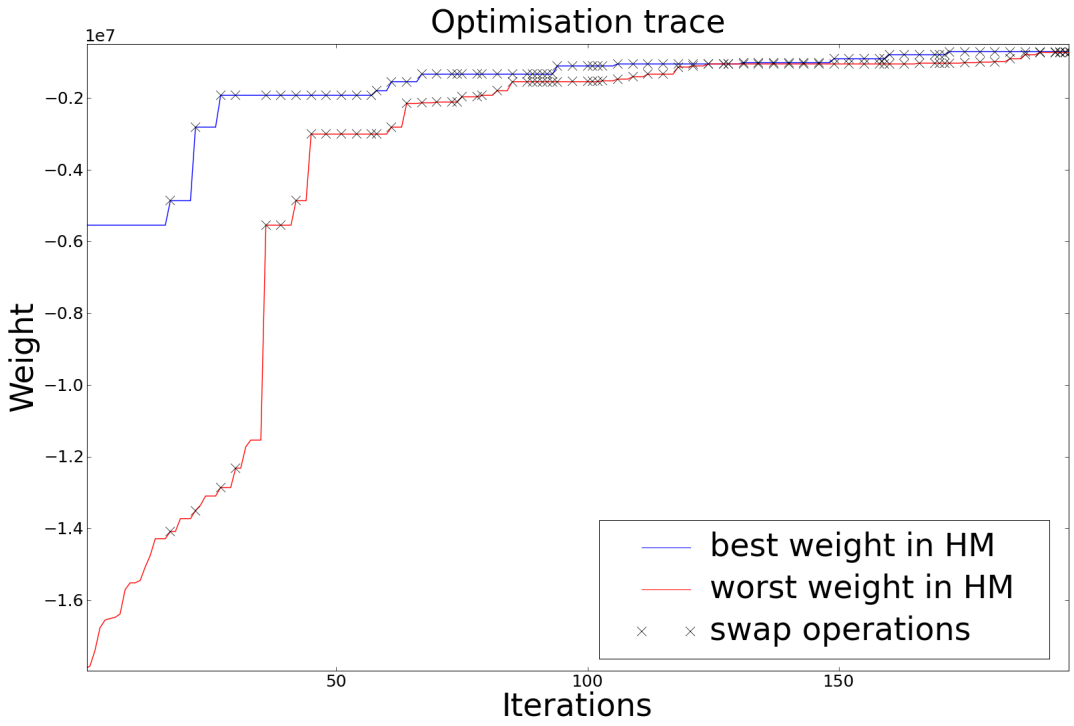

Figure 4.7: The optimisation process is shown through the evolution of the best and worst solution in the HM. Most of the optimisation is done in the first 200 iterations shown here. The crosses indicate iterations in which swap operations occurred and show its importance to successful optimisation.

Unfortunately, memory consideration is not able to maintain a diverse solution set in the harmony memory for long. Notice from the graph that as the worst solution and the best solution become more and more equal, the optimiser's ability to improve significantly on the best solution diminishes. Normally, when the worst solution in the

HM becomes close to equal the best one it is an indication of convergence and that an optimum has been reached. I found that this was rarely the case with DCS. While the best candidate solution reached after 200 iterations is much better than the prior initialisation, the solution is still suboptimal and still has some features incorrectly matched. This means that the HM now consists of several slightly modified copies of a local optimum. Due to the loss of diversity in the HM, further progress and possible escape from this local optimum is unlikely.

However, DCS performs very well over this first few iterations and compares well with other popular algorithms like RANSAC and graph cut. In a real world setting it is very difficult to find the optimal correspondence set mainly due to the imperfect modelling of image features. These limitations in feature extraction and modelling was explored in Section 4.2.1.

4.5 Discussion and Chapter Conclusion

The DCS algorithm shows that harmony search can be adapted to solve labelling problems. Moreover, DCS proved to be very efficient in improving the quality of a naive initial estimate of the correct correspondence set. Even using as few as 50 iterations, DCS was able to find a correspondence set that is 15 times better than the original estimate as measured by the objective function of Algorithm .7.

From the results in the previous section we see that DCS is limited by an early loss of diversity in the HM which leads to a tendency for premature convergence to sub-optimal solutions. This limitation mainly comes from the way the swap operation is introduced as a way to generate solutions that would be impossible for standard harmony search. Unfortunately, the side effect of this is that the carefully designed relationship that pitch adjustment and memory consideration have in exploiting good solutions in the HM while simultaneously maintaining diversity, is lost.

Harmony search is designed to balance exploitation of previous good solutions and exploration of the search space by carefully managing the use of three methods for generating improvisations namely, memory consideration, pitch adjustment and random selection. This balance between exploration and exploitation is an important feature in all heuristic optimisation algorithms and this aspect of harmony search was thoroughly explored in Sections 2.4 and 2.6. Pitch adjustment works together with memory

consideration since the starting value to be pitch adjusted is chosen using memory consideration. When pitch adjustment is removed the balance between exploitation and exploration is upset which results in the tendency to over-exploit leading to premature convergence as seen in DCS.

During the design of DCS it was hoped that this balance could be restored by lowering the HMCR to encourage more random selection and therefore exploration of the search space. However, experimentation into this showed that lowering the HMCR causes DCS to quickly degrade into something that performs only marginally better than random search. While it may not get stuck in local optima as easily as before, the extremely slow convergence that results make this an impractical solution to the problem.

Remember that DCS is essentially solving a labelling problem and that the components of the solution vector are not values that can be incremented or decremented to find neighbouring solutions. We investigated the most important repercussions of this in Section 4.3.1 but there is another more subtle effect that could explain the relatively poor performance of harmony search when applied to labelling problems.

Memory consideration and pitch adjustment both rely on independent components in the search space that can be optimised independently. We have already shown that due to the uniqueness constraint this assumption is not valid in the visual correspondence search problem. The swap operation was introduced as a method that does not rely on this assumption and could work with memory consideration to act as the exploitation part of the optimiser. However the swap operation does not have the elegant and cooperative relationship with memory consideration that pitch adjustment has and, as we have seen, what tends to happen is that exploitation is primarily done by the swap operation which leaves exploration of the search space to memory consideration.

While memory consideration was modified (see Algorithm .4) to take into account the uniqueness constraint in an attempt to relax the independent components assumption, it also introduced an unnatural bias. The special set, $\mathbf{V}$, that ensures that the uniqueness constraint is not violated constrains memory consideration to only those vector in the HM that have component values not present in $\mathbf{V}$. This problem is aggravated by the fact that solution vectors are ordered. When a new vector is improvised the first components are set before the ones at the back of the vector. When the first component value is chosen $\mathbf{V}$ is empty and memory consideration can function

normally but as the vector is constructed component-by-component **V** fills up constraining memory consideration more and more. The last components of the solution vector are always biased towards being chosen by random selection from what is left in **V** simply because it becomes highly unlikely that memory consideration will choose a value that is present in **V**. This arbitrary bias can cause DCS to fixate its exploration of the search space around variations of the first few components in the prior with little exploration of the last components leading to suboptimal solutions.

With the arguments raised in the previous paragraphs one could argue that harmony search is unsuitable for the solving of labelling problems. I argue that this is probably true but also that labelling problems like the visual correspondence problem can still be solved using harmony search based algorithms if better or more suitable ways of modelling a solution vector are used. By transforming the labelling problem into one that can be described with a vector of real numbers a more standard form of harmony search can be used to solve this problem. For example, harmony search was successfully used as a dynamic fuzzy clustering algorithm which is also a pixel labelling problem [51]. The authors accomplished this by not representing solution vectors as a set of pixel labels but rather as a vector of coordinates that represent the centre of pixel clusters. In this way the components of the solution vectors are real numbers that can be independently optimised and incremented or decremented to discover neighbouring solutions.

A possible way of transforming the visual correspondence problem in a similar way is to represent the solution vectors as a series of relative motion vectors that map corresponding features to each other. This approach has not yet been explored but considering that relative motion vectors are already used in the objective function of DCS, this could be valuable future research. A potential objective function for this approach would be to project each feature in the reference image using the motion vectors in the solution vector and then choosing the feature in the alternate image that is closest to the projected result as the corresponding feature. A threshold would be needed to decide when the projected feature is to far from any feature in the alternate image to be a possible match leading to an unmatched feature. This approach would not need the swap operation since we are no longer dealing with feature labels and therefore the pitch adjustment operator can remain as part of the algorithm. This

means that the core of harmony search is left unmodified leading to greater confidence in success based on previous research.

I believe that the greatest value of the DCS algorithm is not in its practical usage but in the knowledge gained from designing and experimenting with harmony search as a method of solving labelling problems. Several insights concerning the important relationship between memory consideration and pitch adjustment was discovered through the developing of the swap operation and its coordination with memory consideration. The dual role that memory consideration has in that it can both be responsible for exploitation and exploration of the search space was also thoroughly explored for the first time.

However, I believe that DCS can still have some practical use as well. The optimisation of the prior over the first few iterations is very promising and greatly improves on it. Considering how much improvement was made in as little as 20 iterations DCS could be used as a computationally efficient initialiser for more robust correspondence matching algorithms. RANSAC, for example, performs much better with a good initial estimate of the correspondence set. By initialising RANSAC with only a few iterations of DCS, an algorithm that is both fast and robust can be constructed.

5

Blind Deconvolution of Binary Images

```
Theories are like nets:  he who casts, captures ...
```

– *L. Wittgenstein*

In the previous two chapters harmony search was used to solve two very different problems in computer vision. We saw that the harmony filter, based on harmony search, was very successful in solving the visual tracking problem and managed to outperform many current state-of-the art algorithms. The harmony filter acted on a relatively standard search space and little modification of the original algorithm was needed to adapt harmony search for visual tracking. The novelty of the harmony filter came not so much from the way harmony search was adapted, but rather how it was implemented and how the visual tracking problem was interpreted as an optimisation problem.

However, in the chapter that followed, harmony search was used to solve a very different problem and more dramatic adaptations to harmony search were needed. These included removing one of the three core operations in harmony search and replacing it with one of my own design. While this change had the desired effect of enabling harmony search to optimise in a very different type of search space from that which it was designed for, the resulting algorithm did not perform as well as hoped.

In this chapter harmony search is again dramatically adapted to solve a problem in computer vision that is not usually solved using optimisation algorithms. Two related algorithms are developed and discussed for solving the blind deconvolution problem.

In computer vision the blind deconvolution problem refers to the recovery of an image degraded by an unknown blurring function and random noise. When this process is applied specifically to binary images a unique combinatorial non-convex optimisation problem is created. The search space in this problem is similar to the one encountered in the visual correspondence problem of Chapter 4 and has many of the same problems. Moreover, it can also be, and often is, interpreted as a labelling problem. Referring to the arguments of the previous chapter, it seems likely that the poor performance of DCS will be seen again if one attempts to use harmony search for solving this kind of problem.

However, as we will see in the sections that follow, a different strategy is used this time and while some of the changes to harmony search are quite significant, the core components are left unchanged. The result is an algorithm that is able to achieve 100% accuracy in recovering severly degraded images without any prior knowledge of the source of the degradation.

5.1 Problem Statement

The restoration of bi-level or binary images is important in many applications. Printed text documents, line art, handwritten signatures, vehicle number plates and bar codes are only some examples of images that are ideally binary in nature. Machine vision systems that work with binary images for automatic recognition and identification require that the images be denoised and deblurred prior to processing for best performance. For example, augmented reality systems and camera calibration systems often use binary image patterns as targets. Another example is found in text recognition systems used in computer aided surveillance.

Image blur can originate from many different sources. Out of focus blur is caused when all or some of the image falls outside the lens' depth of field. This causes light from a singe point source to be spread out in the image and appear as a blurred area in the image. Another common cause of image blurring is caused by motion of the camera during exposure. This is known as motion blur and can take many forms depending on the direction and distance of the motion as well as the duration of exposure. The source of the blur is commonly modelled as a blurring kernel that is applied to the source image resulting in the blurred image. This blurring kernel is most commonly

referred to as the point spread function (PSF) since it describes how points (pixels) in the original image are spread (smeared) to form the blurred result.

Unfortunately, all imaging systems also add some noise to the final image which exacerbates the blur problem by adding unpredictable elements and causing blur removal to become an ill conditioned problem [118, 119]. The convolution process represented by equation (5.2) is a common way of modelling this degradation due to blurring and noise.

$$g(x,y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} [f(x-i, y-j)h(i,j)] + n(x,y) \tag{5.1}$$

$$= f(x,y) * h(x,y) + n(x,y) \tag{5.2}$$

where h is the *point spread function* (PSF) that represents the shape of the blur, f is the uncorrupted image and g is the captured image. The system noise is modelled by adding a Gaussian noise term represented by $n(x,y)$.I use $*$ to represent the convolution operator. The process of recovering f from g is known as deconvolution since the degradation due to blurring is modelled as the convolution of the original image and the PSF [119].

The aim of deconvolution is the recovery of the image and when the PSF can be accurately estimated beforehand, deconvolution is made much simpler. This is known as non-blind deconvolution as opposed to blind deconvolution where the PSF cannot be estimated beforehand and must be discovered simultaneously with the recovery of the image. Blind deconvolution is a much more difficult problem and perfect recovery of the image is rare, especially when the image is also corrupted by large amounts of noise.

In the blind deconvolution problem h is not known or cannot be easily estimated *a priori*. Most solutions to this problem involve iterative techniques that estimate f and h by modelling them as parameters to an iterative optimisation problem [11, 118, 120–122]. An alternate approach is to model f and h statistically by assuming that they fit certain distributions. *Maximum likelihood* techniques are then used to maximise the probability of the distribution in an attempt to find the optimal approximation [123, 124].

When working with binary images one would expect that blind deconvolution would be simpler since the search space of possible variations of f and h is much smaller.

It turns out that adding this restriction to the set of conditions in the optimisation problem is very difficult [125]. In most optimisation algorithms it is very useful (and sometimes critical) that the gradient of the objective function used to evaluate solutions, be available. Restricting solutions to binary values makes it very difficult to construct an objective function for which gradient information is available. Chan et al. shows that objective functions of this type defined over the set of binary images are inevitably non convex and convergence to a global optimal solution can therefore not be guaranteed in general [126].

In this chapter I introduce two algorithms based on harmony search that can solve the blind deconvolution problem on binary images and are often able to find the optimal solution. While it is not guaranteed that the global optimum solution will be found, perfect recovery of the image is still possible even when the PSF was not recovered completely. The first algorithm I introduce, CHS, is successful in that it is able to recover the original image but it suffers from a major limitation. I show then that by properly understanding the search mechanism that harmony search uses to find solutions in this unusual search space, this limitation can be lifted. The resulting algorithm is not only more robust and powerful in that it lacks the limitations of CHS, but it also turns out to be much faster. This algorithm, called LEFHS, is the second algorithm introduced in this chapter and represents convincing evidence that, with proper adaptation, harmony search is capable of solving labelling problems.

5.2 Overview of Current Approaches

Numerous techniques have been developed for the blind deconvolution of images [120, 127] but most of these rely on a colour gradient in the images and perform poorly on binary images [124, 128]. Existing algorithms that are specific to the restoration of binary images are generally based on positive semi-definite programming, iterated quadratic programming and other methods that take advantage of the combinatorial nature of the problem [128, 129]. These methods are all based on optimising a global energy function over binary variables. This idea and its relation to other similar problems like binary partitioning and perceptual grouping was studied in detail by Keuchel et al. [125].

5.2.1 Graph Cuts

In Section 4.2.2.2 we saw that graph cut minimisation is an efficient method for optimising the kind of energy functions often found in computer vision. As we will see throughout this chapter, image deconvolution can also be modelled as the minimisation of a properly constructed energy function. However, as previously noted these energy (or objective) functions are non-convex in general making them very difficult to optimise and making it impossible to guarantee that the global best solution is found. In this section, and those that follow, we will investigate some popular ways that researchers have used to overcome the problem of non-convex energy functions.

Raj et al. aimed to design an optimisation algorithm based on graph cut minimisation that is able to minimise a general energy function of the following form [130].

$$||g(x,y) - Hf(x,y)||^2 + G(f(x,y)) \tag{5.3}$$

This is a regularisation of an energy function based on Equation (5.2). $f(x,y)$ and $g(x,y)$ are defined as in Equation (5.2) and H is a matrix that represents the convolution with $h(x,y)$. Since the noise term $n(x,y)$ is not taken into account and since $||g(x,y) - Hf(x,y)||^2$ on its own makes an ill-posed optimisation problem, the energy function is regularised by adding the smoothness term $G(f(x,y))$. This is very similar to the situation that we discussed in Section 4.2.2.2 and the smoothness term here has the same function as when used in the visual correspondence problem.

As in Section 4.2.2.2 the smoothness term can either impose a global smoothness on the image or be some form of a discontinuity preserving term. A global smoothing term makes for an easier to optimise energy function and many algorithms have been developed to minimise this type of energy function [131]. However global smoothness is not realistic in image reconstruction and is therefore not appropriate for the deconvolution problem. Images generally consist of locally smooth areas separated by sharp discontinuities between the object boundaries. A globally smooth minimiser tends to erode these sharp discontinuities which ends up destroying fine detail in the reconstructed image.

However, Raj et al. demonstrated that existing graph cut minimisation techniques cannot be applied to a general energy function of the form in Equation 5.3 when G is discontinuity preserving. They showed that graph cut minimisation can only be applied

in the special case when H is diagonal. However, many problems in computer vision, like image deconvolution, can only be accurately modelled using a non-diagonal H.

Their solution to this problem was to develop a new technique that allows existing graph minimisation algorithms to minimise energy functions with an arbitrary H. Their approach is to dynamically construct an approximation to the original energy function that can be minimised using graph cut minimisation. Even though one cannot prove that the minimum of the approximation represents the minimum of the original energy function, Raj et al. demonstrate that improving the score of their approximation also improves the score of the original.

Their main application for this algorithm was the reconstruction of medical MRI images from the equivalent Fourier data. In this case H encodes the Fourier transform and can be accurately estimated from the dynamics of the MR scanner used to capture the Fourier data. This problem is therefore not blind deconvolution since H is known and neither is it binary. It does however illustrate the difficulties that researchers face when trying to optimise energy functions that originate from the image deconvolution problem. These problems only become worse when H is unknown and when f consists of binary labels instead of brightness intensities.

5.2.2 Positive Semi-Definite Programming

Many researchers have realised that binary image deconvolution is both important to many applications in computer vision and is not a problem that can simply be seen as a degenerate form of intensity image deconvolution. Shen et al. proposed an algorithm based on positive semi-definite programming that is able to perform blind deconvolution on binary images [132].

They recognised that the global energy function associated with blind deconvolution is non-convex and that minimisation of this function would be very difficult. Furthermore, even if it could be minimised one could never guarantee that the global minima was reached. Their approach was therefore to construct a convex relaxation of the original energy function that can be minimised using positive semi-definite programming (PSD).

Shen et al. models the deconvolution problem as a binary combinatorial optimisation problem that explicitly takes the binary nature of the original image into account.

The argument is as follows. Since the pixels in f are binary each pixel in g must originate from either one of two prototypical values, u_1 and u_2. Without loss of generality let $u_1 = -1$ and $u_2 = 1$. To restore the binary vector $f = \{f_1, f_2, \ldots\}$ from the observed image g, the following objective function must be minimised:

$$z(f) = \frac{1}{4}\sum_i((u_2 - u_1)f_i + u_2 + u_1 - 2g_i)^2 + \lambda \sum_{<i,j>} (f_i - f_j)^2 \ , \tag{5.4}$$

where λ is a smoothness term parameter, and $< i, j >$ is the set of adjacent pixels centred around f_i. This objective function has a similar form to the one investigated in Section 4.2.2.2, specifically the graph cut energy function of Equation (4.2). This function also has a data fitting (also called the goodness of fit) term and a smoothness term.

The fist term, $\frac{1}{4}\sum_i((u_2 - u_1)f_i + u_2 + u_1 - 2g_i)^2$, is the data fitting term and is minimised when each value of f, either u_1 or u_2, is set to correspond correctly with the observed values in g. The second term, $\lambda \sum_{<i,j>}(f_i - f_j)^2$, is the smoothing term that encourages local smoothness around a point in the reconstructed image. Through a process called positive semi-definite relaxation this function is then arranged into an approximation that is optimisable using PSD (see [132] for deatils).

The performance of this algorithm was tested on various 10×10 binary images that were synthetically blurred with various PSFs. The main reason that these examples are kept small is because the computational time grows exponentially with the number of pixels in the image. On a 3.2 GHz PC with 512MB RAM this algorithm was able to deconvolve a 10×10 binary image that was blurred with a 3×3 non-binary PSF in 22 seconds. The image was also corrupted with noise by adding Gaussian noise with a variance of 0.01. With this amount of noise and with a relatively large PSF compared with the size of the image, the accuracy of the reconstructed image was only 75%. With smaller PSFs the accuracy can be as high as 91%.

The main limitation, however, is clearly the computational complexity that grows exponentially with the size of the images. It is also worth mentioning that even though this approach is currently not applicable to blind deconvolution, blind deconvolution was mentioned as future research of the PSD approach.

5.2.3 Iterated Quadratic Programming

Another approach that also stemmed from the realisation that the binary nature of the problem should be explicitly modelled uses an iterated quadratic programming technique to optimise a restoration filter designed to recover the original binary image [128].

As with the previous two algorithms that we investigated, the first step is to overcome the non-convex energy function of Equation (5.3) and arrange it into a form that is optimisable. In this case the energy function must be made convex as the quadratic programming technique can only minimise convex energy functions.

The problem is formulated as follows. Starting with $f(x, y)$ as the original uncorrupted image, let $\hat{f}(x, y)$ be the reconstructed binary image that results from the convolution of the observed image, $g(x, y)$, with a restoration filter, $w(x, y)$. Without loss of generality, let the two binary values be -1 and 1. A properly designed restoration filter would then construct $\hat{f}(x, y)$ such that $(\hat{f}(x, y))^2 - 1 \approx 0$. The minimisation problem is then defined as follows:

$$\text{minimise } ||t(x, y)||^2 \tag{5.5}$$

$$\text{subject to } -t(x, y) \preceq [w(x, y) * g(x, y)] \bullet [w(x, y) * g(x, y)] - 1 \preceq t(x, y) \tag{5.6}$$

where $\preceq$ denotes element-by-element comparison and $\bullet$ denotes element-by-element multiplication. The auxiliary variable $t(x, y)$ is introduced to convert the problem into a standard form that can be optimised using quadratic programming. The constraint in Equation (5.6) causes the minimisation of $t(x, y)$ to tune $w(x, y)$ to produce only binary outputs. Notice that a small value for $t(x, y)$ is only obtained when $[w(x, y) * g(x, y)] \bullet [w(x, y) * g(x, y)] - 1 \approx 0$ which means $\hat{f}(x, y) \approx \pm 1$. However, the constraint is not convex and so a recursive filtering technique must be used since there is no way to directly calculate the global optimal solution.

This algorithm is able to process much larger images and a 256×100 pixel text image was deconvolved successfully in [128]. Unfortunately, no indication of the accuracy is given in this publication and the blur added to the image is very mild. Only judging visually, the accuracy appears to be no better than 80% (see Figure 5 in [128]). However, the author mentions that further testing with more severe blurring is a goal of future research.

5.3 Counterpoint Harmony Search

The counterpoint harmony search (CHS) algorithm was my first attempt at using harmony search to solve the blind deconvolution problem for binary images. As we saw in the previous sections, this is a difficult combinatorial optimisation problem over a vector of binary labels. Since the calculation of the global optimum was also shown to be NP-hard the problem is intractable and some approximation has to be found [125].

We also saw that restricting the problem to binary images does not simplify the problem like one would expect, but instead adds to the difficulty of the problem [125]. By restricting the problem to binary values the energy function inevitably becomes non-convex which usually means that some approximation to the energy function must be constructed and optimised instead [133].

CHS is based on the harmony search algorithm and is able to recover a binary image from a non-binary source image that was heavily corrupted by noise and blur [27]. It is able to do this by using a modified harmony search algorithm to find the minimum of an objective function that is based on the regularised energy function of Equation (5.3). The energy function is regularised to be discontinuity preserving which is essential when working with binary images since sharp discontinuities between black and white pixels are the norm. While it is impossible to guarantee that the global optimum of such an objective function will be found during optimisation without exhaustive search (the problem is NP-hard), I demonstrate in several examples that CHS is often able to recover the original binary image to 100% accuracy even when global optimum of the energy function was not found.

Before giving an overview of CHS some extra notation is introduced in the next section that will be used in the remainder of this chapter. I start with a new formulation of Equation (5.3) that then forms the basis of the objective function used in CHS. *Total variation regularisation* is also introduced as a way of regularising the objective function while still being discontinuity preserving.

5.3.1 Objective Function

Going back to Equation (5.2) one sees that $g(x, y)$ should ideally be equal to $f(x, y) * h(x, y)$ and would be were it not for noise. Therefore it seems reasonable to evaluate candidate solutions of f and h by convolving them and comparing the result with g

(this is the same argument used to reach Equation (5.3)). Let $\hat{f}_i$ and $\hat{h}_i$ be the estimate of f and h from the ith iteration of the improvisation process. I then define

$$\hat{g}_i = \hat{f}_i * \hat{h}_i \tag{5.7}$$

$$\mathcal{F}(\hat{f}_i, \hat{h}_i) = ||\hat{g}_i - g||^2_{\Omega_h} \ , \tag{5.8}$$

where $\mathcal{F}$ is the objective function and Ω_h is the region in h in which it is known to be active (which is often just the PSF size). Ω_h is also known as the *support* of the PSF and can usually be estimated from the captured image. By minimising $\mathcal{F}$ over the set of all $\hat{f}$ and $\hat{h}$, one can find the best estimate for f and h, but only if one assumea that the effect of noise on g is small.

Unfortunately, it is known that the convolution process tends to amplify noise, making $\mathcal{F}$ ill-conditioned [118, 119]. It is also known that minimising $\mathcal{F}$ is in general an ill-posed problem making it necessary to regularise and constrain $\mathcal{F}$ into a form that can be minimised towards an optimum [118]. This was also the purpose of the smoothing term in Equation (5.3).

One of the most successful approaches to regularising $\mathcal{F}$ is based on the *total variation* of an image and is defined as

$$||f||_{tv} = \sum_{\Omega_y} \sum_{\Omega_x} |f'(x, y)| dxdy \tag{5.9}$$

where Ω_x and Ω_y define the support of the image and $|f'(x, y)|$ is the gradient magnitude. The method is known as total variation regularisation (TVR) and is known to suppress noise while maintaining details in *blocky* images [126, 134, 135].

Adding the TVR terms to the fitness function then yields

$$\mathcal{F}(\hat{f}_i, \hat{h}_i) = ||\hat{g}_i - g||^2_{\Omega_h} + \alpha||\hat{f}_i||_{tv} + \beta||\hat{h}_i||_{tv} \ , \tag{5.10}$$

where α and β are Lagrange multipliers that balance the focus of the fitness function between suppressing noise and maintaining a reasonable goodness-of-fit with the captured image. A good way to choose α and β was introduced by Chan et al[121]. They show that choosing α directly proportional to the amount of noise in g and β directly proportional to the severity of the blur gives best results. I follow the same approach

in CHS and found that α and β can be kept constant for almost all examples once acceptable values for them have been found based on a single generic example problem.

As shown in the previous sections, this is by no means the only method to regularise the fitness function and we will see in the next section that using only TVR is not always recommended. For example, when image blur is caused by some more exotic PSFs, like motion blur at a 45° angle, TVR tends to blur away the sharp edges in the PSF causing slow convergence.

5.3.2 CHS Overview

The original harmony search algorithm avoids local optima by maintaining an element of random search throughout the optimisation process. The harmony memory consideration rate (HMCR) controls the optimiser's tendency to perform random search or to converge around minima stored in the harmony memory [4]. This method has shown to be successful in escaping local minima but initial testing showed that it has difficulty finding the global minimum when the number of local minima is very large compared to the size of the search space. The objective function not only has many local optima densely packed throughout the search space but the global minimum can also be present at more than one location in the search space [118]. This means that there is more than one combination of f and h that convolves to form g even in the absence of noise. This means that even if the objective function could be perfectly optimised it still does not ensure that the correct solution was found. This situation is illustrated in Figure 5.1.

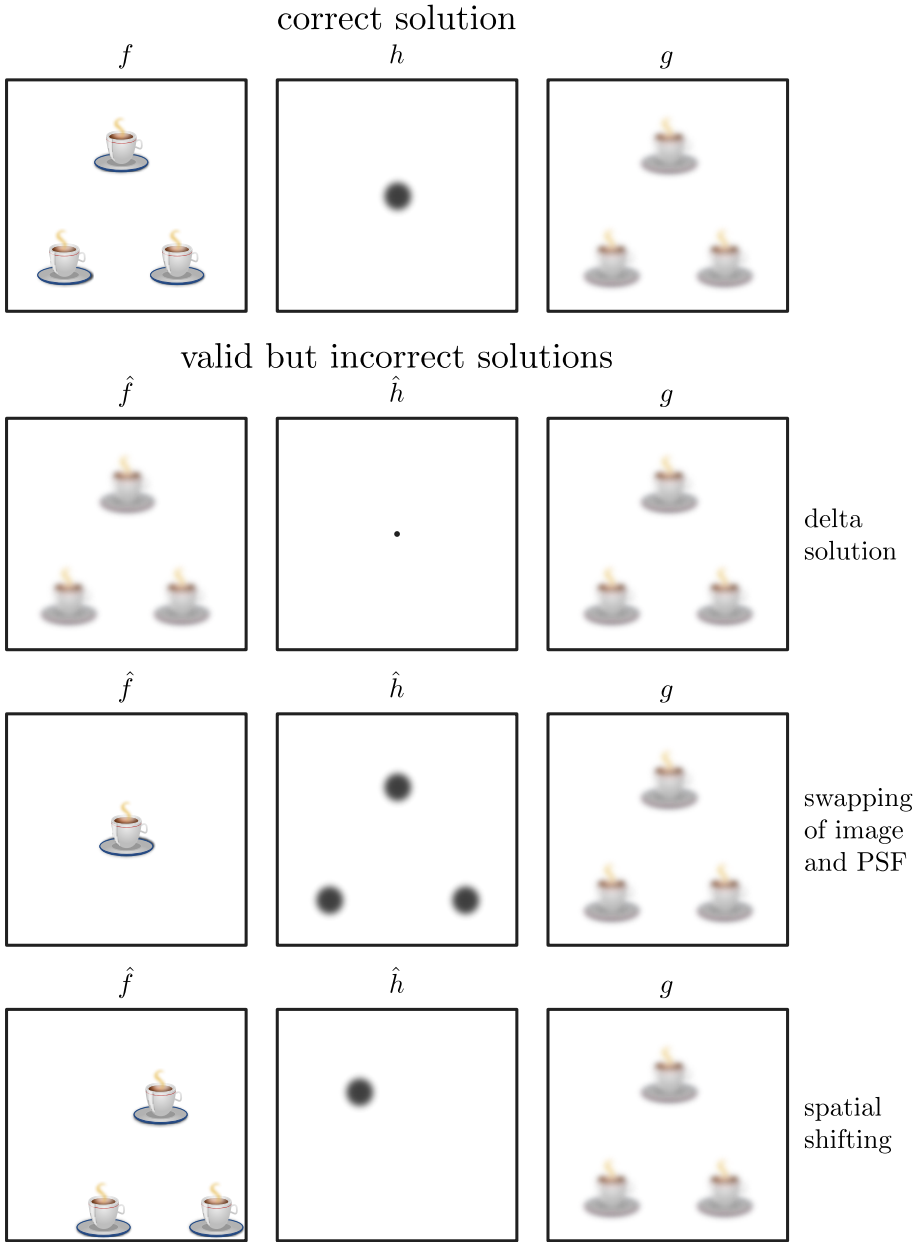

Figure 5.1: This diagram illustrates that blind deconvolution is inherently ambiguous when defined as a valid solution to Equation 5.2. In the first row an image and PSF are given that results in the blurred image in the last column. In the rows that follow various combinations of different images and PSFs are given that result in exactly the same blurred image, making them equally valid solutions even though none of them represent the original unblurred image. Three examples of how such alternate solutions may be generated are given namely, using the delta solution, swapping the roles of the image and the PSF and spatially shifting both the PSF and the image in equal and opposite directions. There are other possibilities and these can be combined to make even more complex examples resulting in an almost limitless number of alternate solutions.

In this situation we need an alternate approach to escaping local minima and also to guide the optimiser towards what will most likely be the real solution. The inspiration for CHS comes from a method used in genetic algorithms when the number of local optima is too large for the mutation operator to avoid them. The population of solutions is then often split up into multiple independent groups or *islands* that are independently optimised. Genetic algorithms that use this method are known as island model parallel genetic algorithms.

Island model parallel genetic algorithms converge to multiple local optima simultaneously by dividing the population of possible solutions into islands that converge independently from each other [136, 137]. It is this idea that inspired CHS that divides the harmony memory into multiple islands that also converge independently.

The five main steps of CHS is very similar to a generic harmony search optimiser with the main difference being the division of the HM and the addition of one extra step called the *cadenza operation*. An overview of the main steps in CHS are as follows.

1. Initialise the harmony memory and split it into S equal size islands.
2. Randomly pick one member from one of the islands in the harmony memory. Its island becomes the active island.
3. Use this member to derive a new improvisation using a method called the cadenza step (in a way this replaces the improvisation step).
4. Update the population of the active island with the new improvisation.
5. Repeat steps 2-4 until convergence is detected.

However, splitting the HM into multiple independent islands would still not solve the ambiguous solutions problem of Figure 5.1. This is why we need a unique way of improvising new solutions in step 3, which I call the cadenza operation (hence the name *Counterpoint* Harmony Search).

5.3.3 The Cadenza Operation

To illustrate the CHS algorithm the simple example of Figure 5.2 is explained in a step-by-step way. It illustrates the first iteration in an implementation of CHS where

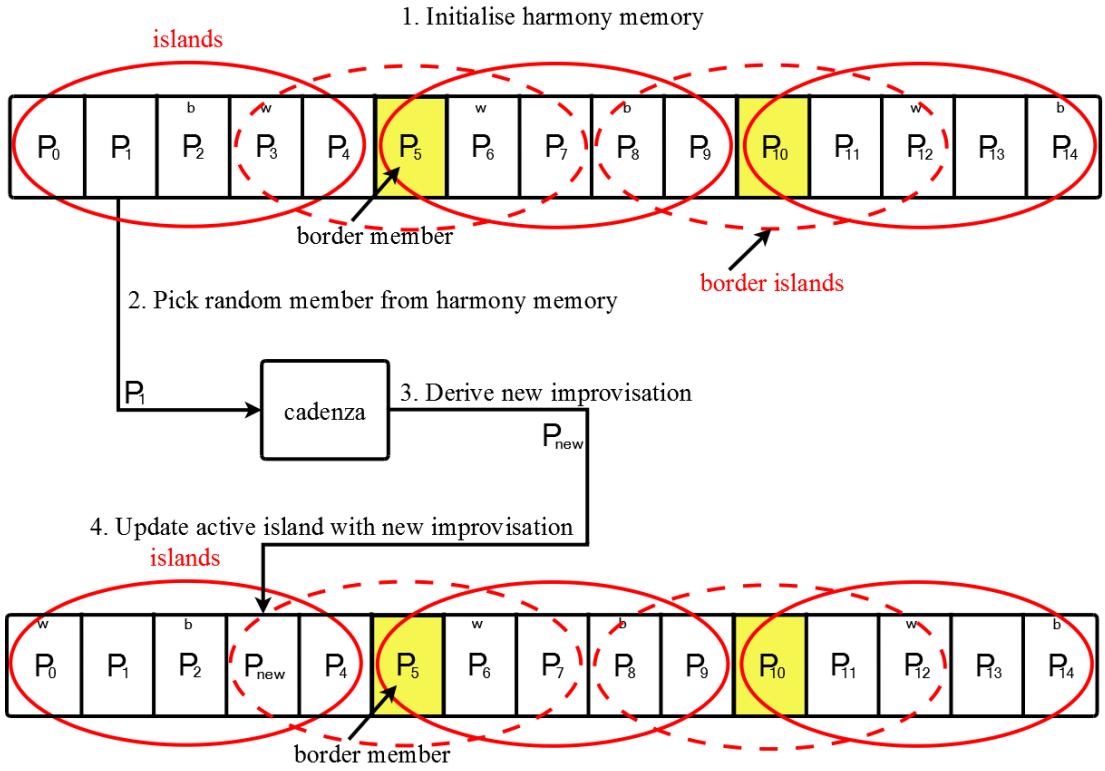

Figure 5.2: This diagram illustrates how the CHS algorithm works through a simple example.

the island size (IS) was chosen to be 5 and the harmony memory size (HMS) was chosen to be 15 which means that the harmony memory consists of 3 islands.

In the first step the harmony memory is initialised with 15 sample solution vectors. Each solution vector is a concatenation of the elements of f and h and therefore only contains binary values. A diagram illustrating the layout of the HM is shown in Figure 5.3.

Diversity in the harmony memory is very important as this prevents multiple islands converging to the same local minima. Initial solution vectors are therefore generated so that diversity is maximised while still talking into account any prior knowledge that is available. In this case the only prior knowledge is that noisy solutions should be suppressed and that the recovered image should have some similarity with the blurred one. I also make the initial assumption that the PSF support is centred around middle of the PSF image and consider this as prior information on the PSF. Initialisation is

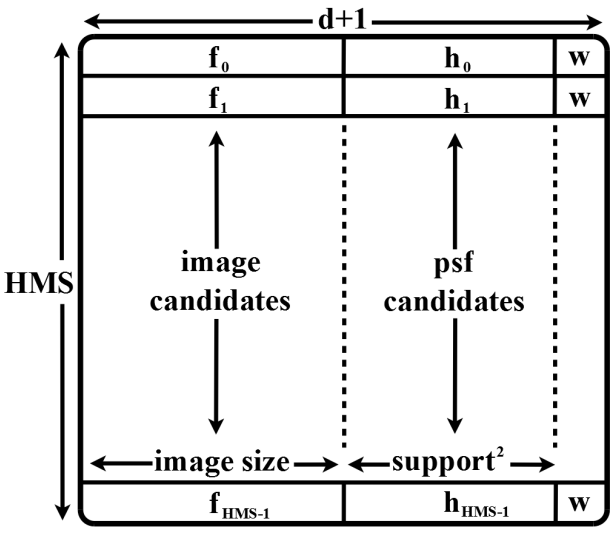

Figure 5.3: The HM is essentially divided into two halves. The first half of each candidate solution vector (represented by rows) is the recovered image ($\hat{f}$) while the second half is the PSF ($\hat{h}$). The final component of each vector is the fitness weight as measured by the objective function.

therefore divided into the separate initialisation of f and h based on some assumed prior knowledge that usually proves to be accurate even though CHS is not dependent on the validity of these assumptions.

Since g is a blurred version of f it seems logical to base initial estimates of f on g. We therefore use the following formula to generate initial estimates of f.

$$\hat{f}_0(x,y) = \begin{cases} 0 & \text{if } g(x,y) + \rho < 0.5 \ , \\ 1 & \text{if } g(x,y) + \rho \geq 0.5 \end{cases} \tag{5.11}$$

where ρ is a zero-centred Gaussian random variable. This creates diverse random solutions for $\hat{f}$ based on thresholded versions of g.

The $\hat{h}$ part of the initial solutions are all initialised to be identical and it is left to the cadenza operation, which is discussed later, to maintain diversity. I initialise $\hat{h}$ to be the smallest possible PSF that is not considered as noise by the cadenza operation. It will become clear later when the cadenza operation is discussed, that the following 2×2 PSF meets this criterion.

$$\hat{h}_0 = \frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

An illustration of this PSF is given in Figure 5.4. The small square in the centre is the support of the PSF and is the reference point for further improvisation using the cadenza operation.

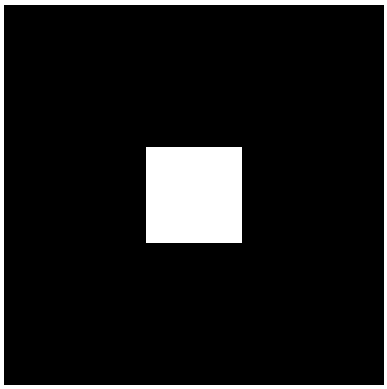

Figure 5.4: This is the initial estimate of the PSF and serves as the reference point for the cadenza operation.

Once all the vectors of the HM has been initialised, each vector is evaluated using the objective function $\mathcal{F}$. The vectors with the best and worst fitness values are identified and indicated by the letters b and w respectively in Figure 5.2. This is used to monitor the evolution of each island and is also used when updating the harmony memory after improvisation.

In Figure 5.2 the HM is shown as a row vector at the top and bottom of the figure. One must understand that each member of this vector is itself a vector, namely a candidate solution vector. The 15 solution vectors, P_0 – P_{14}, are then divided into three equal sized islands as indicated by the red ovals.

The second step is to randomly pick one candidate solution vector from the HM to base an improvisation on. The island of which this vector is a member then becomes the active island. In the example of Figure 5.2 the P_1 vector was chosen so the first island becomes the active one.

The P_1 vector is then used as the basis for a new improvisation by applying a set of rules to a small number of its pixel elements. This improvisation through a set of specialised rules is called the cadenza operation. A small number of pixels (approximately 1% of the total) is chosen from both the $\hat{f}$ and $\hat{h}$ part of the vector and the following steps are applied to each of them.

1. Count the number of white pixels in a 3×3 area centred around the pixel under consideration.
2. If all nine pixels are white leave the pixel white.
3. If all nine pixels are black leave the pixel black.
4. If any other situation flip the colour of the pixel (white becomes black and black becomes white).

The $\hat{h}$ and $\hat{f}$ parts are considered separately and when a pixel changes colour from one part the other part is prevented from changing as well. This is to prevent oscillation between two similar points in the search space that causes slow convergence in the final iterations. The cadenza operation prevents noise from being generated by the optimiser while generating diverse solution vectors based on existing vectors in the harmony memory. An example image with the pixels highlighted that could be potentially colour flipped by the cadenza operation is shown in Figure 5.5. This illustrates that areas of importance, which were initially identified by the blurred input during initialisation, can dynamically grow or shrink without introducing noise into areas that showed no activity in the original blurred input.

Once an improvisation has been generated by the cadenza operation it gets evaluated by the objective function and the HM is updated. If the fitness of the new improvisation is better than the fitness of the worst vector in the active island the worst vector is replaced by the new improvisation and all members of the active island are re-evaluated to determine the best and the worst vectors. In the example of Figure 5.2 the new improvisation, P_{new}, had a higher fitness than P_3 which was the worst member in the first island. P_{new} therefore replaced P_3 and the vectors of the first island were re-evaluated.

A special case occurs when an improvisation is based on a border member. A border member is defined as the first vector in each island except for the very first island and is represented by the yellow highlighted vectors in Figure 5.2. When an improvisation is based on a border member the active island is defined as the first half of the island that the border member is a member of and the second half of the preceding island. These new temporary islands are therefore a combination of the vectors of two neighbouring islands and are indicated by the dotted red ovals in Figure 5.2. In this way good

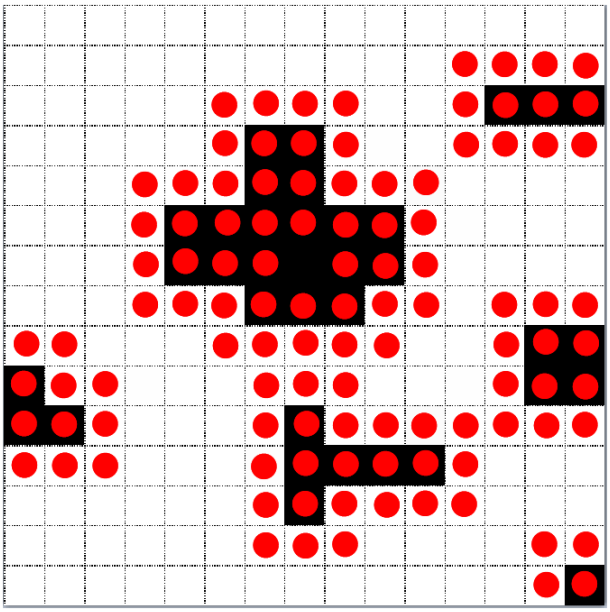

Figure 5.5: The cadenza operation flips the colour of selected pixels based on a set of rules. The pixels that can potentially flip colour are indicated by red dots.

solutions from one island can affect neighbouring islands improving the overall quality of the HM with a minimal loss of diversity. This special ability that border members have is similar to the ability of certain members in the island model parallel genetic algorithm that migrate to neighbouring islands to serve the same purpose.

This process of generating new improvisations and updating the memory is repeated until the best vector in one of the islands is such that convolving $\hat{f}$ and $\hat{h}$ results in a perfect match of g or until the maximum number of iterations is reached. Due to noise in g a perfect match with g is very rarely found even when $\hat{f}$ and $\hat{h}$ are equal to f and h. In practice the optimisation is only stopped when the maximum number of iterations is reached or when the difference between $\hat{g}$ and g is sufficiently small.

5.3.4 Improving the Objective Function

In Section 5.3.1 I explained how the objective function is regularised using TVR. When using binary images the total variation of the image is simply a measure of the number of black-to-white pixel transitions or edges in the image. Due to the way we initialise the PSF and generate new PSFs using the cadenza step, TVR tends to cause square

shaped PSFs to be generated much more frequently than other PSF shapes. This becomes a problem when the correct PSF shape is not rectangular as is commonly the case with motion blur. To combat this bias we need to adjust the objective function slightly so that square PSFs are not favoured as much.

Figure 5.6 illustrates how the current form of the objective function places an unfair bias on certain PSF shapes. Two PSFs are shown, on the left a 4×4 square PSF and on the right a PSF that represents motion blur at a $45°$ angle. The problem is that both these PSFs have the same number of edges even though the square represents more growth from the initial PSF. Therefore both of these PSFs will have equal total variation which can suppress the exploration of certain PSF shapes. This is somewhat counter-intuitive since one would expect the objective function to not favour any one of the two examples over the other due to their total variation being equal.

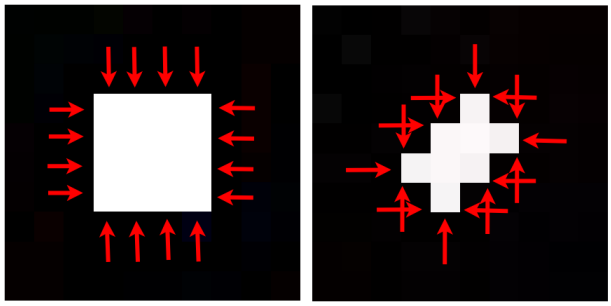

Figure 5.6: Two PSFs with equal total variation. The red arrows indicate the edges and are both equal to 16 in these examples.

However, the cadenza step tends to grow the PSF out from the centre towards the edges of the PSF support equally in all directions. This is mostly due to the way PSFs are initialised. Since the PSF is initialised as a square and since squares have the lowest total variation for a given number of pixels, it tends to grow into larger squares more often than other shapes. I discourage this tendency towards square PSFs by introducing another regularisation term into the objective function that simply counts the number of white pixels in the PSF. This is known as total energy regularisation (TER) and the new objective function then becomes

$$\mathcal{F}(\hat{f}_i, \hat{h}_i) = ||\hat{g}_i - g||^2_{\Omega_h} + \alpha||\hat{f}_i||_{tv} + \beta||\hat{h}_i||_{tv} + \beta||\hat{h}_i||_{te} \ , \tag{5.12}$$

where $||\hat{h}_i||_{te}$ indicates the total energy of the PSF. A new Lagrange multiplier could be introduced for the total energy of $\hat{h}$ but I found that the introduction of a new parameter only complicates the objective function and that one Lagrange multiplier for both the total variation and the total energy works acceptably well.

This additional regularisation term penalises a candidate PSF for every pixel of growth independent of the resulting total variation. The total energy also grows faster than the total variation which amplifies its effect as the PSF grows larger and more complex. This is exactly the behaviour that is needed since we still need the noise preserving quality of TVR to be most prominent at the start of the optimisation process when the PSF is still small and the generation of noise is more likely. As the PSF grows the effect of TVR becomes less prominent compared to TER which is insensitive to the PSF shape.

However, one should notice that both TER and TVR favour small PSFs over large ones. This is a very valuable feature since we are simultaneously trying to recover the image, f, and having a large PSF early in the optimisation process will blur together all the detail that might otherwise be discovered in f. Ideally, the optimiser should almost exhaustively search through all possible small PSFs before moving on to larger possibilities since this will give it more time to recover a rough outline of the features in f.

5.3.5 CHS Results

In this section I evaluate selected examples of blind deconvolution of various binary images using the CHS algorithm. The first example is the 24×12 text image of Figure 5.7. The original image is convolved with a PSF estimated to have a support area of 5×5 and no noise is added to the result. Notice that the support size of the PSF does not have to be known prior to recovery as this is true blind deconvolution. However, one can usually estimate a rough expected size of the PSF simply by visual inspection of the blurred input. By giving CHS a rough estimate (which may be greatly exaggerated) of the size of the PSF, convergence may be accelerated.

This example was deconvolved using the CHS algorithm with the following parameters. The HMS was chosen as 99 with an island size of 9 which means that the harmony memory consists of 11 islands. The two Lagrange multipliers, α and β were chosen as 0.005 and 0.003 respectively. As can be seen from Figure 5.7 100% accuracy

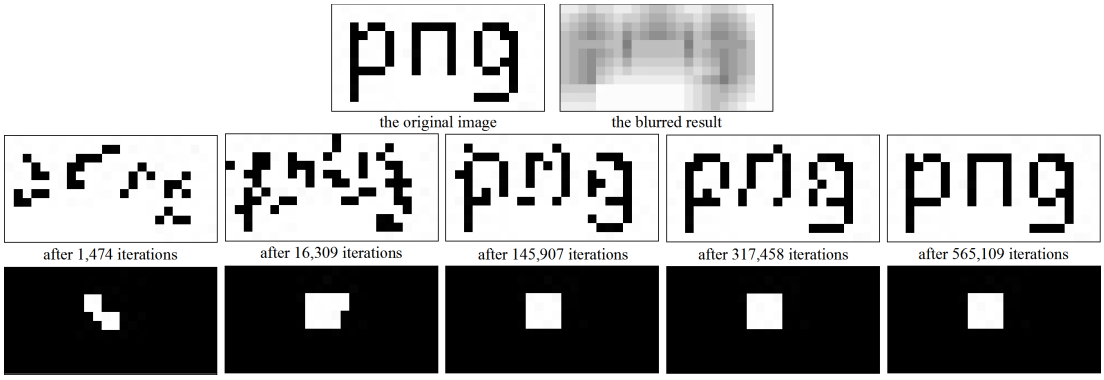

Figure 5.7: This is a simple example of binary blind deconvolution using CHS. The top row shows the original image and the blurred result. The bottom two rows show the progression of the deconvolved image and the progression of the PSF respectively.

was achieved after 565, 109 iterations. This took 32 seconds on my Intel i7 920 based test machine (1 core used at 3.4GHz). The CHS algorithm identfied the unknown PSF as a 4×4 square blur that is slightly off centre towards the top left. Notice that the support of the PSF was only slightly overestimated but that this did not significantly affect convergence speed.

In the second example a more exotic PSF is used and some noise is added to the blurred input. In Figure 5.8 we see the 10×10 text image before and after convolution with the PSF and additional zero-centred Gaussian noise with a standard deviation of 0.01 added.

I used the CHS algorithm with the same parameter values as in the first example but increased the value of α to 0.008 in an attempt to speed up convergence on this smaller image. This example converged much quicker than the previous example and 100% accuracy was achieved after only 4 seconds. As we can see from Figure 5.8 the PSF was identified as a skew motion blur at a 45° angle and that the support was again only slightly overestimated.

The third example was constructed to be the most challenging of the three. An 18×16 binary image was constructed to contain a large number of fine detail and sharp edges. These fine edges are most affected by blurring and are easily lost when even the smallest PSF is applied. After convolution with a PSF I added five times the amount of noise than in the previous example. The original image and the noisy blurred result is shown in Figure 5.9.

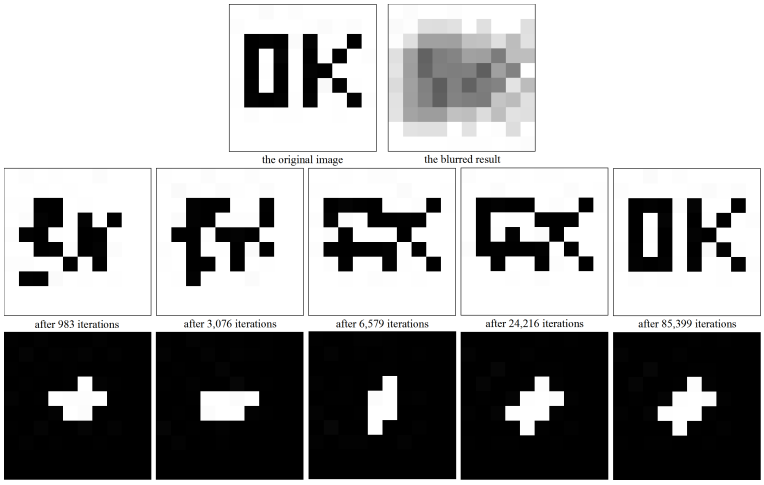

Figure 5.8: This is the second example of binary deconvolution using CHS. The top row shows the original image and the blurred result. The bottom two rows show the progression of the deconvolved image and the progression of the PSF respectively.

I again used the CHS algorithm to deconvolve the example using the same parameter values as in the first example. From Figure 5.9 we see that CHS was able to completely deconvolve the image to 100% accuracy even with the added noise. 100% accuarcy was achieved after $768,038$ iterations and took 39 seconds on the same test machine as in the first example. In this example the PSF was identified as a 2×4 vertical motion blur slightly shifted towards the top of the image.

In all three my examples 100% accuracy was achieved in less than a minute of processing. Different image sizes, different PSF shapes and different amounts of noise were used to show that the CHS algorithm is robust in a number of different situations.

I now compare these results with those found by Shen et al. [129, 132]. They design a convex fitness function that is then optimised using positive semidefinite programming (see Section 5.2.2). By segmenting the image into smaller overlapping blocks this approach overcomes the huge computational cost of deconvolving large images. This allows them to deconvolve larger images and successful deconvolution of 100×100 images have been shown [129, 132]. However the accuracy of this method is only 98.8% on the best example. This accuracy falls below 88% when Gaussian nose with standard deviation of 0.01 is introduced (the same as my second example). Shen et al. notes that deconvolved text becomes difficult to read at accuracies below 90%.

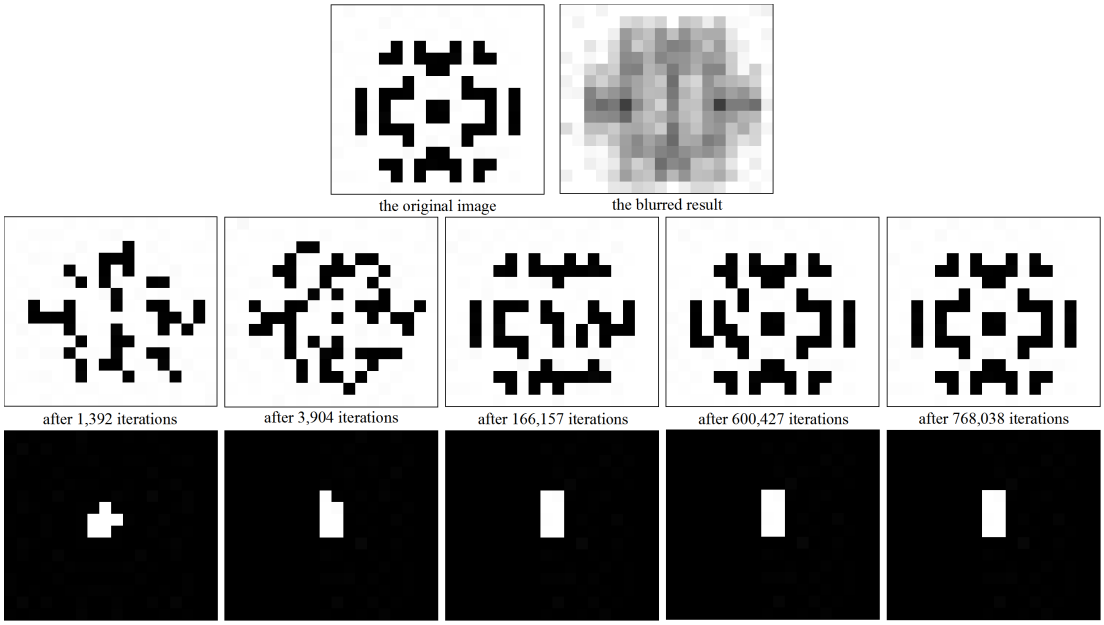

Figure 5.9: In this example CHS is used to deconvolve an image containing containing fine detail that could easily be incorrectly recognised as noise. The top row shows the original image and the blurred result. The bottom two rows show the progression of the deconvolved image and the progression of the PSF respectively.

In another example Shen et al. uses their algorithm to deconvolve a 10×10 randomly generated binary image. A 3×3 uniform PSF (similar to the 4×4 PSF of my first example) is then used to blur this image. The support of this PSF is very large compared to the size of the original image. When a PSF with a support of 3×3 is used to blur a 10×10 image, severe blurring would result and we expect the accuracy of the deconvolution algorithm to drop. As expected the accuracy dropped to 77% for noiseless deconvolution. When Gaussian noise with standard deviation of 0.01 is added to the blurred image, the accuracy drops to 75%.

In the CHS 10×10 example (see Figure 5.8) I add the same amount of noise but use an even larger PSF with a support size of 4×4. Using this PSF, the size of the PSF support is 16% of blured image's size. Even with this large PSF and added noise CHS manages to deconvolve the image to 100% accuracy. As we saw in the third example, the CHS algorithm can even cope with five times the amount of noise and still produce 100% accurate results.

5.4 The LEFHS Algorithm

While the results from CHS was initially impressive especially when compared to other current work, it is still much slower than most other algorithms when recovering images of similar size. But the biggest limitation of CHS is that it is only able to recover binary PSFs and images corrupted by binary PSFs. While there are many applications for the recovery of strictly binary images, PSF are not binary in general. Most real-world blurring can only be accurately modelled with non-binary PSFs. Therefore, for a deconvolution algorithm to be practically useful it must be able to recover images blurred by non-binary PSFs.

The Largest Error First Harmony Search (LEFHS) algorithm was developed to address this limitation and create a harmony search based blind deconvolution algorithm for binary images that is practically usable [28].

5.4.1 LEFHS Overview

The LEFHS algorithm is based on the CHS algorithm and is similar to it in many respects. Like CHS, it also escapes from local optima by dividing the HM into islands, but uses a different method to combine solutions from different islands. Instead of using the cadenza operator to construct new improvisations based on one member from the HM, LEFHS uses a local search for the image part of the improvisation, and the original harmony search improvisation for the PSF part. These changes allow LEFHS to improvise non-binary PSFs. The objective function is also modified to better accommodate non-binary PSFs. In the paragraphs that follow I introduce LEFHS and focus particularly on the differences between it and CHS.

Like CHS, LEFHS starts with the initialisation of the HM. The image and PSF parts are initialised as in CHS (see Section 5.3.3) leading to an HM that is represented by a matrix of solution vectors as illustrated in Figure 5.3. As with CHS, the number of decision variables, or the dimension (d) of a solution vector, is the sum of the number of pixels in the image and the square of the PSF support size. The PSF support size can optionally be provided as a parameter to the algorithm but when not known is initialised to 2 and slowly increased during optimisation until a maximum size is reached.

Once initialised, the HM is divided into S islands and the following steps, explained in the paragraphs that follow, are repeated until convergence.

1. Randomly pick one member from the active island as the basis for a new improvisation (the active island is initialised to the first island).
2. Alternate between exploring the image or PSF part of the improvisation.
3. If the image part is explored build the error map and use a local search method to discover new candidate images. If the PSF part is explored use the original harmony search improvisation scheme to discover candidate PSFs.
4. Add a fitness weight to this improvisation by evaluating with $\mathcal{F}(\hat{f}_i, \hat{h}_i)$.
5. If the weight is better than the worst weight in the active island, replace the member that has the worst weight with the new improvisation. If not, then consider this an idle iteration and do nothing.
6. If the maximum number of consecutive idle iterations are reached for the active island continue on to the next island and make it the active island. If this is the last island replace the worst member from all islands with a new improvisation made by combining the best member from all islands with the worst member from all islands and set the active island back to the first one.

The primary differences between CHS and LEFHS are found in steps 3,4 and 6. In step 3 a local search based on the error map is used to quickly guide the search process to the most likely solution, greatly improving the speed of convergence. In step 4 a modified objective function, designed for non-binary PSFs, is used to evaluate candidate solutions. In step 6 a novel scheme is used to ensure a steady increase in the quality of the HM while simultaneously maintaining the diversity in the separate islands. These steps are discussed in detail in the paragraphs that follow.

5.4.2 The LEFHS Improvisation Scheme

In step 2 of the above overview improvisation of the restored image and the PSF are alternated instead of simultaneously optimised. I found that concentrating on one part while keeping the other set to a previously known good solution results in faster

convergence than when both are simultaneously optimised. When both the PSF and image are optimised simultaneously the solutions tends to oscillate around an optimum instead of converging to it.

Candidate PSFs are improvised using the standard harmony search scheme of Chapter 2. The fret width (FW) parameter represents the granularity or the number of allowable grey values in the PSF. For example, a FW of 256 would mean that each pixel in the PSF can be one of 256 distinct values which is the maximum for an 8-bit pixel. However, perfect recovery of the PSF is not required for perfect recovery of the image and restricting the PSF to a smaller granularity speeds up convergence. It is therefore recommended that the FW parameter be kept small (< 25). This step is not sensitive to the exact values of the HMCR and PAR parameters and they can be chosen equal to the values recommended in harmony search literature [4, 5]. Let `rand` be a random variable uniformly distributed between 0 and 1. Pseudo code for the improvisation of a new PSF is then as follows.

```
for each pixel in the PSF do
  if rand <= HMCR
    mem = randomly chosen member from the active island
    pixel = mem's corresponding value
    if rand <= PAR
      randomly add or subtract FW from pixel
  else
    pixel = randomly chosen from allowed range
```

This is almost exactly like the original harmony search improvisation step. The only difference lies in the selection of the pixel value when the HM is considered. Instead of choosing a solution vector from the whole HM only members from the active island is considered. This ensures that members from different islands cannot influence each other and therefore acts to maintain diversity of solutions in the HM.

However, a much more efficient method is used to improvise candidate images that take into account the way PSFs of different support sizes act on images. Candidate images are improvised by building an error map based on the blurred image input and the current improvisation initialised from the active island. Let e_i be the error map, $\hat{f}_i$ and $\hat{h}_i$ be the candidate image and PSF from the improvisation and g the blurred image. Each pixel of the error map is then calculated as follows.

$$e_i(x, y) = (\ g(x, y) - \hat{f}_i(x, y) * \hat{h}_i(x, y)\ )^2 \tag{5.13}$$

From equation (5.2) one expects that perfect solutions for $\hat{f}$ and $\hat{h}$ would give the smallest values in e and that the largest values in e would indicate incorrect areas in f. I therefore limit the alteration of $\hat{f}_i$ to an area bounded by the size of the PSF around the centre of maximum error (e_{max}) which is defined as the pixel in $\hat{f}_i$ corresponding to the maximum error value in e_i. The search area is bounded by the size of the PSF since it determines which pixels contribute to the error. Only pixels in a border around e_{max} half the support size of the PSF, can contribute towards the error in $\hat{g}_i$. This is illustrated in Figure 5.10.

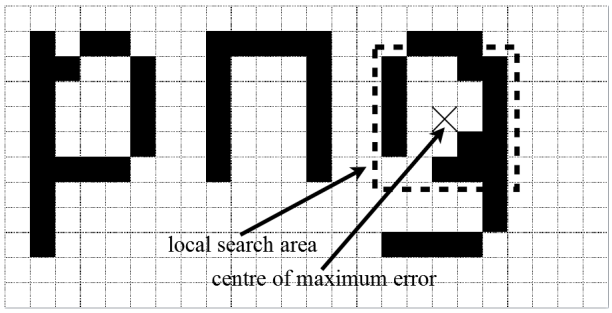

Figure 5.10: In this example the PSF support size is 4. Therefore, the local search area around e_{max} is a 5×5 area with e_{max} at its centre.

By concentrating the search in an area that is most likely to contain the most incorrect pixels and making the resulting search space as small as possible, both the speed and accuracy of the optimiser are greatly improved. It is also interesting to notice that the support size of the PSF now has a very direct effect on the speed of the optimiser. When the PSF is small the search area around the centre of maximum error is also small leading to quicker convergence. However, the search area grows quadratically with the support size of the PSF. This is one of the reasons for bounding the support of the PSF to the smallest possible value at the beginning of the search and only growing to larger PSFs when no further improvements to the best solution is made.

5.4.3 Normalised Regularisation

Next in step 4, a weight is associated with the improvisation by evaluation with an objective function similar to equation (5.10). Like the CHS algorithm I add an extra regularisation term but found that TER in non-binary PSFs tends to grow them too

large, too quickly leading to local optima. Due to the new improvisation scheme it is essential that the PSF be kept as small as possible for as long as possible during improvisation. To do this the regularisation term is normalised so that pixels far from the PSF centre contribute more than pixels close to the centre. This tends to keep the PSFs smaller for longer giving the optimiser more time to explore candidate images with smaller PSFs before moving on to larger ones. The improved objective function is based on equation (5.10) with the addition of a normalised TER term added defined as follows:

$$||\hat{h}_i||_{\text{normTE}} = \sum_{\Omega_h} t(x,y)\sqrt{(x_0-x)^2+(y_0-y)^2}$$

where

$$\begin{aligned} t(x,y) &= \begin{cases} 0 & \text{if } \hat{h}_i(x,y) = 0 \ , \\ 1 & \text{if } \hat{h}_i(x,y) > 0 \end{cases} \\ (x_0,y_0) &= \text{ the centre of the PSF} \\ \Omega_h &= \text{ the support area of the PSF} \end{aligned}$$

5.4.4 Improved Crossover from Individual Islands

Finally in the last step, the active island is updated based on the number of consecutive idle iterations. The maximum number of idle iterations is a parameter that controls the optimiser's emphasis towards maintaining population diversity opposed to speed of convergence. The value should be chosen so that the active island is only updated when no further progress is made in the current one. Once all islands reach this point a new improvisation is made by combining the best and worst ($m_{best}(x,y)$ and $m_{worst}(x,y)$) members from all islands as follows:

$$i_{\text{new}}(x,y) = \begin{cases} m_{\text{best}}(x,y) & \text{if rand} > \tau \ , \\ m_{\text{worst}}(x,y) & \text{if rand} \leq \tau \ , \end{cases} \tag{5.14}$$

where rand is a random variable uniformly distributed between 0 and 1, and τ is a parameter that shifts the focus between emphasising the already good solution $m_{\text{best}}(x,y)$ which might be a local optima, and adding diversity from $m_{\text{worst}}(x,y)$ which is almost certainly a local optima. I found the best results when τ is kept small to favour

$m_{\text{best}}(x, y)$ and kept $\tau = 0.3$ in all my experiments. This method is then used to update all the members of the worst island. The idle iteration counter is then reset for all islands and optimisation returns to the first island that is then made the active island again. This process repeats until convergence is detected or the maximum number of iterations is reached.

The effect that this crossover scheme has on the evolution of the HM is illustrated in Figure 5.11. The top graph shows the global situation with the best solution in the HM indicated by a blue line and the worst indicated by a red line. The first thing to notice is that the weight of the best solution is a strictly growing function while the weight of the worst solution is not. This is highly unusual for an algorithm based on harmony search and one would expect that both the best and worst solutions of the HM would only improve with increasing iterations and certainly never worsen. one would expect that any worsening of the weight would indicate divergence which should be impossible since the HM is only updated with solutions that improve and replace the current worse one.

However, due to the unique crossover scheme that replaces all the elements of the worst island without taking fitness weights into account, it is possible that the fitness weights in the worst island may actually worsen instead of improve. Since this is the worst island it can only influence the value of the global worst member. This is better illustrated by an enlargement of Figure 5.11 shown in Figure 5.12. As in Figure 5.11 the top graph shows the global situation over all islands and the bottom graph shows each island individually. Notice that the point in the top graph where the weight of the worst member suddenly drops (the red line) corresponds to the point where the worst weight of the yellow island (the worst one) also drops. This is the point where the crossover operation was used to replace all the members in the yellow island. Notice also that even though the worst weight (the solid line) worsened further, the best weight in that island improved significantly upgrading the yellow island to becoming the third worst island. More importantly, however, is that when the optimiser returned to the yellow island it managed to improve the solutions from that island to such an extent that it became the best island with all the members becoming better than any of the members in any other island. The diversity that was injected through the crossover operation proved to be exactly what was needed to improve the current global optimal solution.

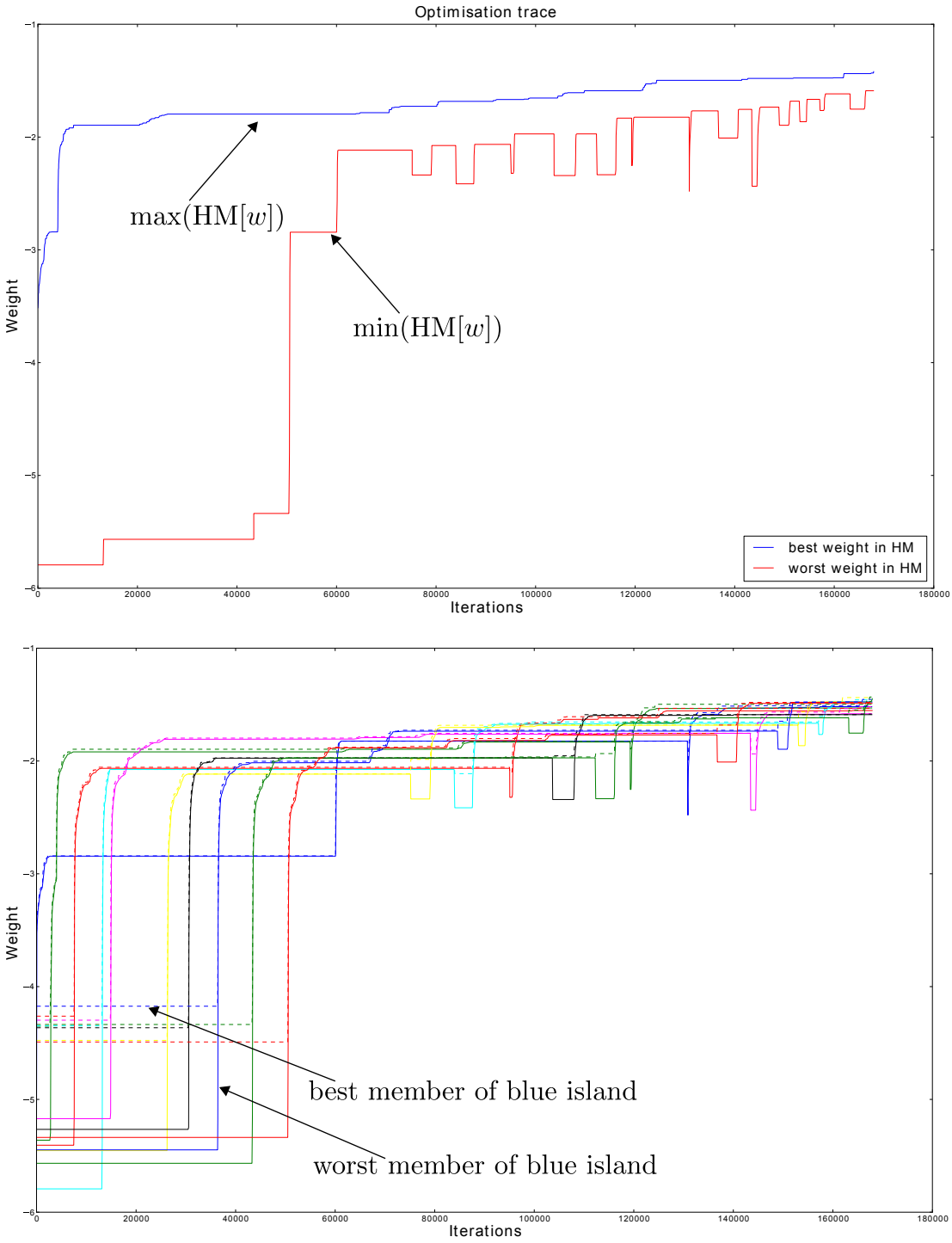

Figure 5.11: This graph shows the evolution of the best and worst members in the HM. In the top graph only the global best and worst fitness weight is given for each iteration over all islands. The bottom graph shows the evolution of each individual island. Each island is identified by its own colour (some islands have to share a colour) and dashed lines indicate the best member of the island while solids lines indicate the worst.

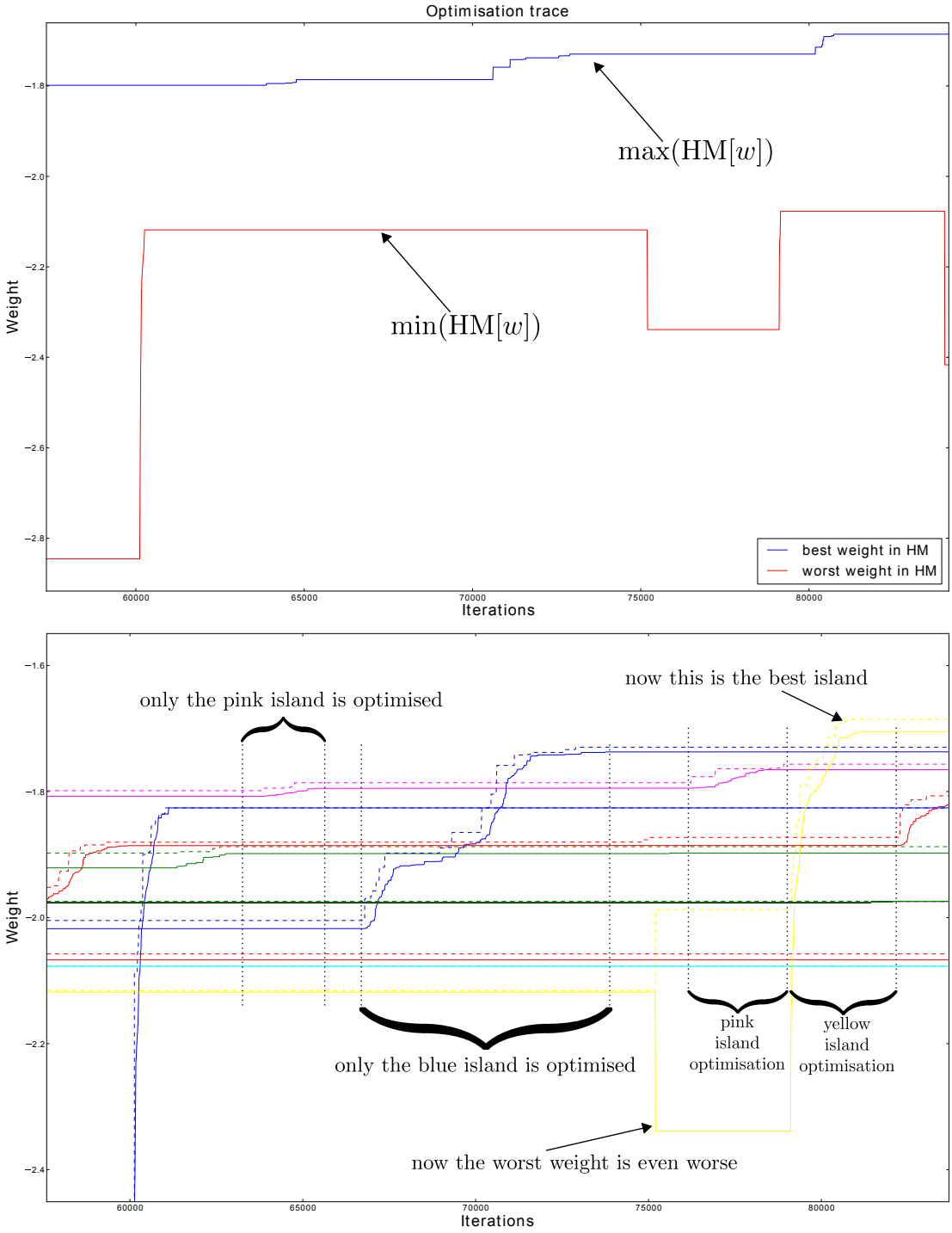

Figure 5.12: This enlargement of the HM evolution clearly shows the independent evolution of each island. The effect that the improved crossover scheme between islands have is also emphasised.

Another interesting feature of LEFHS that Figure 5.12 highlights is the independent optimisation of the islands. Notice the two areas highlighted on the bottom graph. When the pink island became the active island the optimiser was only able to slightly improve the best member even though this was the second best island. Therefore, the optimiser quickly abandoned the search after only a few iterations and moved on to the next island which was the blue one. The blue island, however, was significantly improved eventually becoming the best island even though it started out as the fourth worst one. The optimiser therefore spent many more iterations optimising this island as can be seen from the much larger area emphasised in the graph.

5.5 Results

In this section I illustrate blind binary image deconvolution using the LEFHS algorithm with three examples. I compare accuracy and performance with that of the CHS algorithm and the system developed by Shen et al. [27, 132].

The first two examples are blurred text images created by taking a known binary image, adding zero-mean Gaussian noise and convolving the result with a known PSF. This is the same example that was used to illustrate CHS but this time a different non-binary PSF is used. The advantage of using simulated noise and blur is that challenging examples can be constructed in a controlled way and since the perfect solution is known independently, the accuracy of the algorithm can be measured exactly. The LEFHS parameters, including the Lagrange multipliers in the objective function and the harmony search parameters, are kept constant in all the examples and were not fine-tuned to a specific example. The maximum PSF support size was also kept constant to a 6×6 area in all examples and I always used a 150 member HM divided into 10 islands of 15 members each.

I chose the maximum support size of 6×6 based on the image size of 24×12. My previous experiments during the development of LEFHS and CHS showed that it becomes very difficult to successfully recover an image once it has been corrupted with a PSF larger than 15% of the image size. Once PSFs larger than 15% of the original image are considered recovery starts to become impractical. I therefore choose the maximum support size as the largest $n \times n$ square smaller than 15% of the original

image size. The size and number of the HM islands were also chosen experimentally by trying multiple values and keeping the set that gives the best results.

The first example is the same 24×12 text image containing the letters *png* that was used in Section 5.3.5. Gaussian noise was added with a standard deviation of 0.01 and convolved with a 4×4 non-binary square shaped PSF. After 167,198 iterations the original binary image was recovered exactly and the PSF was recovered with only a one pixel inaccuracy. This results are shown in Figure 5.13.

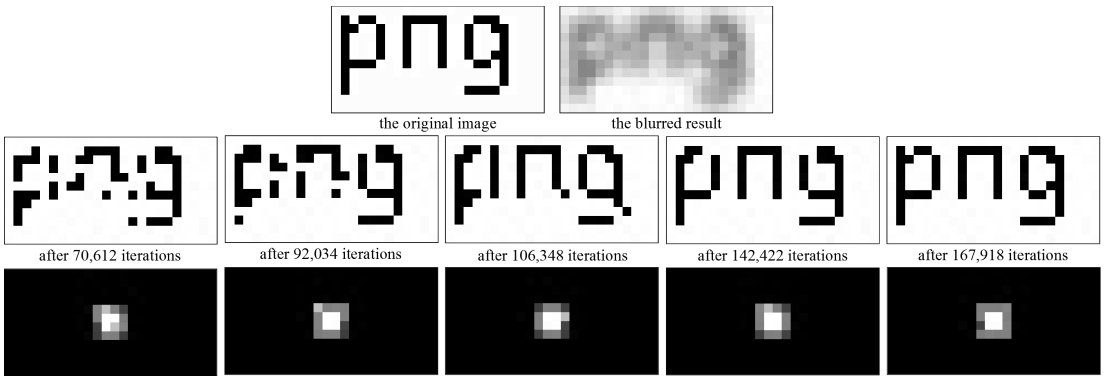

Figure 5.13: In this example perfect restoration of a 24×12 image degraded using a 4×4 PSF and Gaussian noise with a standard deviation of 0.01 is achieved. The two bottom rows show the recovered image and PSF at various iterations.

In the second example I show that LEFHS still performs well even under severe noise by adding 5 times the amount of noise as in the previous example. The same 24×12 text image is used again but this time I add Gaussian noise with a standard deviation of 0.05. I again convolve with a 4×4 PSF but use one with a slightly darker border this time. As seen in Figure 5.14, perfect restoration of the binary image was again accomplished after 105,427 iterations with a three pixel error in the PSF.

In the final example a real image captured with an entry level consumer grade camera (Cannon PowerShot A1100 IS) is restored using the same parameter settings as used in the previous examples. The image was captured using the camera's BW mode with the contrast and sharpness set to maximum. This was done to get the best approximation of a true binary image. The camera was moved upwards during exposure to simulate vertical motion blur. Figure 5.15 shows the original blurred image of the letter A cropped to 12×15 sub-image and the results after only 6,147 iterations. It is difficult to know the exact accuracy of this restoration since the original image is

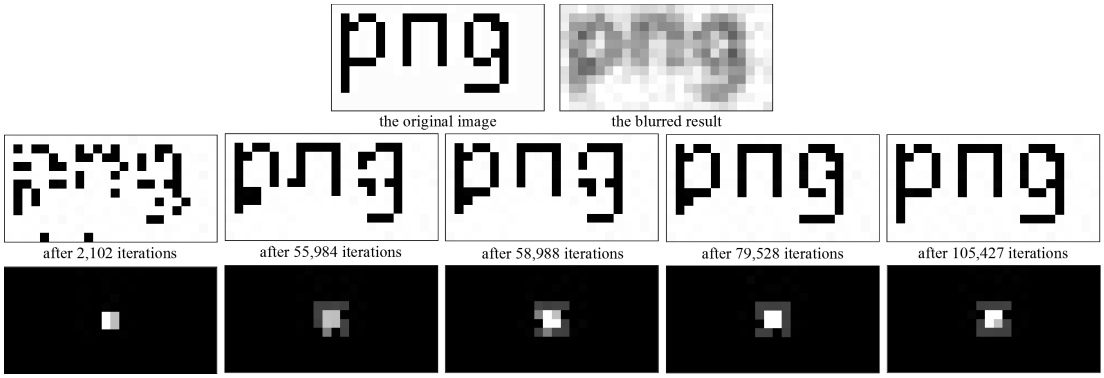

Figure 5.14: In this example perfect restoration of a 24×12 image degraded using a 4×4 PSF and Gaussian noise with a standard deviation of 0.05 is achieved. The two bottom rows show the recovered image and PSF at various iterations.

not known but the letter A is clearly recognisable and the recovered PSF indicates a vertical motion blur as expected.

The image from examples 1 and 2 was specifically chosen since it was also used in reporting results from the CHS algorithm. When Gaussian noise with a standard deviation of 0.01 was used the CHS algorithm took 565,109 iterations to achieve perfect restoration even when restricted to binary PSFs only. In another example the noise is increased to a standard deviation of 0.05 and perfect restoration is reported after 768,038 iterations. From the first two LEFHS examples we see that even when not restricted to binary PSFs, LEFHS achieves the same perfect recovery of the blurred images that CHS does but much faster using dramatically fewer iterations.

We can again compare these results to the system developed by Shen et al. based on positive semi-definite programming [132]. As mentioned in the CHS section, theirs is one of the leading methods and one of the very few that is designed specifically for binary image deconvolution. In their best example a restoration accuracy of 98.8% is reported in a simulated example with no noise added. The accuracy drops to 88% when Gaussain noise with a standard deviation of 0.01 is added. At this accuracy the restored text image becomes difficult to read and an accuracy of 90% is reported as the minimum accuracy for comfortable reading of restored text images [132].

Their accuracy drops further when the PSF is large compared to the image. When a 3×3 PSF is used to blur a 10×10 image the reported accuracy is only 75% when noise with a stardard deviation of 0.01 is added. In their example the PSF support

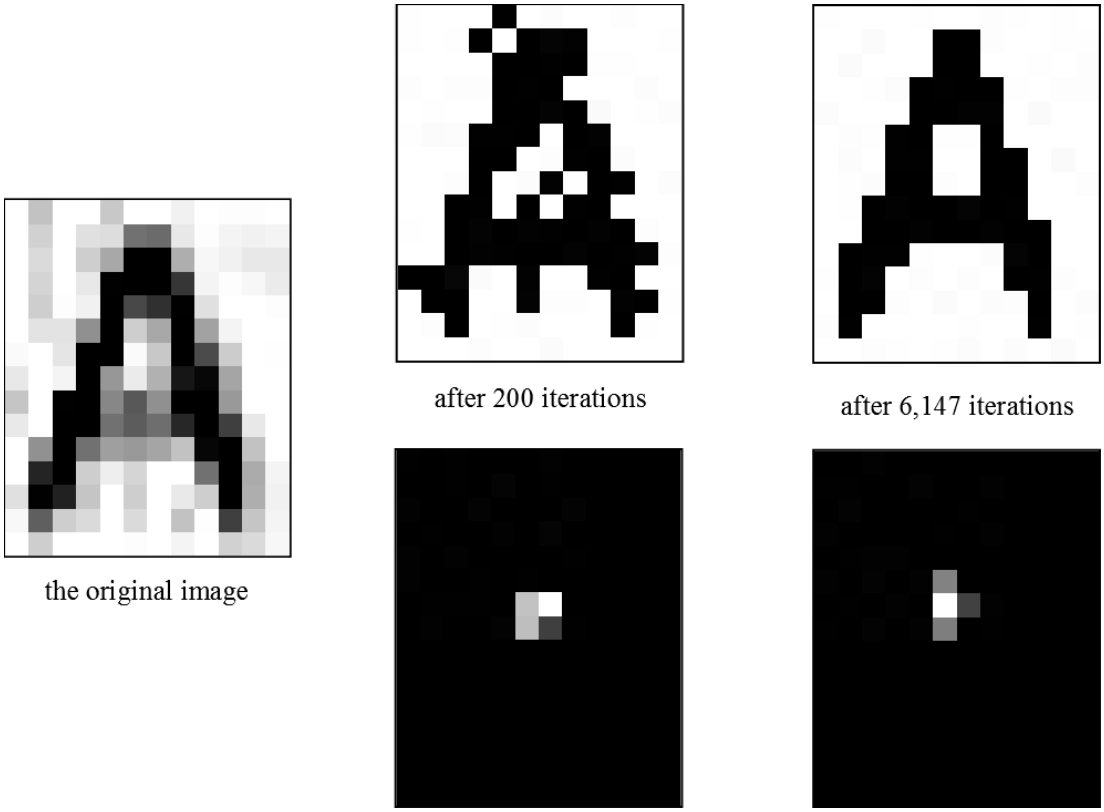

Figure 5.15: In this example a real 12×15 image is recovered using the LEFHS algorithm. Since it was impossible to know the exact accuracy optimisation was stopped after only 6,147 iterations when the letter became clearly identifiable.

size is 9% of the image size compared to the 5.5% ratio used in the first two examples of this section. Even with this large support size LEFHS still achieves 100% accuracy when recovering binary images even when the noise is 5 times larger compared to the results from [132].

5.6 Discussion and Chapter Conclusions

As mentioned in the beginning of this chapter, the focus in this chapter was on showing that harmony search can be successfully adapted to optimise in challenging search spaces where component values are not values that can be incremented or decremented to reach neighbouring solutions. This type of problem is usually modelled as a labelling problem which in this case would be binary labels.

The problems associated with labelling problems were thoroughly discussed in chapter 4 and many of those same problems had to be overcome in the design of CHS and LEFHS. In CHS the cadenza operation was used as a replacement for the improvisation step and proved useful in both suppressing noise and guiding the optimiser to likely good solutions without resorting to pitch adjustment that is useless in labelling problems.

However, the cadenza operation only works in strictly binary situations leading to the main limitation of CHS, namely its inability to recover images blurred by non-binary PSF's. To relax this restriction non-binary candidate PSF had to be allowed in the HM greatly increasing the size of the search space and making the cadenza operation useless. A method needed to be found that could both overcome the enormous search space and be able to improvise increasingly better candidates of the strictly binary original image and the non-binary PSF. Since the primary focus is not to recover the PSF exactly the search space problem was partially solved by restricting the PSF to only a few grey levels and relying on the closest non-binary estimate to be good enough for accurately estimating the original unblurred image.

While this helped to keep the search space to a manageable size, a very efficient method was now needed to improvise image candidates since PSF candidates are now rarely accurate representations of the true PSF. By localising the search to the smallest possible region around a known origin of error I maximise the efficiency of the optimiser where candidate images are concerned. Improvisation of the PSF is controlled by keeping the support size as small as possible for as long as possible and only allowing it to grow when no further improvement is being made to the candidate images. This strategy combined with the fact that candidate values are now non-binary allows for the successful use of the traditional harmony search improvisation step but only when the PSF is being optimised.

In this type of problem the design of the objective function, and more specifically its regularisation is very important. From current literature we saw that the total variation method of regularisation is successful at suppressing noisy solutions while at the same time maintaining fine detail that is often lost by more aggressive regularisation techniques. For this reason, I based the original objective function for CHS on the standard TVR deconvolution energy function (see Section 5.2.1).

However, due to the cadenza operation TVR alone caused the optimiser to favour generating square PSFs which caused convergence to be very slow when the true PSF was not square. An added regularisation term solved this problem at the cost of making the tuning of the objective function more difficult.

The objective function had to again be slightly modified for use in LEFHS to ensure that the support size of the PSF is kept as small as possible. The TER term that was added to prevent the tendency for square PSFs was now further modified by normalising it to penalise larger PSFs more than smaller ones.

By carefully designing the objective function to match the improvisation method and modelling the problem appropriately I showed that harmony search can successfully be adapted for even challenging computer vision problems. The results from the LEFHS algorithm speak for themselves and show that a harmony search based approach can even perform better than algorithms specifically designed for the blind deconvolution of binary images. The only limiting factor at this time to the success of LEFHS is the size of the image that is to be recovered. Due to the quadratic growth of the search space with the size of the image and the size of the PSF, the largest image that can currently be recovered practically using LEFHS is ≈ 300 pixels with a PSF support size no larger than $\approx 6 \times 6$. One should however realise that quadratic growth is still considered good when dealing with NP-hard problems.

One way that future versions of LEFHS might overcome this limitation is by the same method used by Shen et al. to overcome a similar problem. They used a segmentation scheme in larger images that divide the image into small overlapping tiles that are more manageable to recover [129, 132]. Using this method, binary images of 100×100 pixels have been recovered to an acceptable accuracy. The main difficulty of this approach is in the recombination of the separately recovered tiles and the possibly conflicting PSFs between different tiles. If an efficient and accurate tiling scheme can be combined with LEFHS it is possible that much larger images can then be practically recovered leading to an algorithm that is practically usable in more applications.

6

Conclusions

> It's what we learn after we know it all that is important.
>
> – *Earl Weaver, Baltimore Orioles*

6.1 Realisation of the Research Objectives

The main purpose of this thesis was to propose that the harmony search algorithm is a valuable addition to the algorithms and techniques used in computer vision. Harmony search has never been used in any computer vision application before my publication on the harmony filter for image tracking [23]. This introduction to the field of computer vision led to other harmony search based algorithms both by my own research and by other research groups. The success of the harmony filter alone is sufficient evidence that harmony search is at least worth investigating for researchers of computer vision.

The harmony filter was shown to be more robust than the particle filter and UKF implementations that it was tested against and was able to recover a target that was lost due to occlusion and erratic movement in situations where the other implementations could not. Not only is it more robust, but it is also faster. In all my tests the harmony filter was faster than both the UKF and the particle filter and in at least one example it was more than 2 times faster than the UKF and 4 times faster than the particle filter implementation (using 300 particles).

However, the harmony filter is a relatively direct application of harmony search and required little adaptation. To show the true power of harmony search and its

adaptability to many varied applications one would have to apply harmony search to different types of problems found in computer vision. I therefore adapted harmony search to two other problems found in computer vision that demanded much greater adaptation of the original algorithm.

The result of these efforts was the creation of four novel algorithms namely, the harmony filter, DCS, CHS and LEFHS. Of these four all of them showed that harmony search can be adapted to successfully solve the computer vision problem that was assigned to it and three of them proved to perform better than the current state-of-the-art. During the time spent on this study another research group also explored the use of harmony search in computer vision and created a successful harmony search based algorithm for use in image segmentation [51]. This adds further evidence that the introduction of harmony search to computer vision is leading to better algorithms being developed in many computer vision applications and that further harmony search related research will lead to further advances in the computer vision discipline.

CHS and LEFHS were developed to solve the blind deconvolution problem for binary images. This is a particularly challenging problem for optimisation algorithms due to the unique multi-modal objective function associated with binary labelling problems of this type. Both CHS and the improved version called LEFHS showed very impressive performance in solving this problem. Both are able to recover a severely degraded image to 100% accuracy and LEFHS is able to maintain this performance even when arbitrary (non-binary) PSFs are the source of the degradation. LEFHS is able to mainatain 100% recovery accuracy even when the image is severly degraded by large PSFs and significant noise. Under similar conditions the best accuracy reported by another modern algorithm was 75%.

6.2 Further Findings

Perhaps the greatest value of this research is perhaps not in the practical value of the algorithms that were introduced or in the *proof of concept* value that the introduction of harmony search to computer vision gives, but rather in the insights gained during their development. The methods used in adapting harmony search led to many valuable insights into managing large search spaces, efficient convergence detection, recovery from getting stuck in local optima and many others. For example, in the development of

the harmony filter a study was done on the sensitivity of the harmony search parameters to the speed of convergence. This led to a set of parameters that can be considered an optimal starting point when determining the best parameter values for many similar harmony search based adaptations. It also led to insight into which parameters should be adjusted first when optimising performance and how much one can expect to gain from this fine tuning.

From the parameter sensitivity analysis we saw that harmony search is most sensitive to the HMCR and therefore this is the best place to start the fine tuning of the parameter set. The HMS was found to be slightly less sensitive than the HMCR and this would likely be the best candidate for further tuning after the HMCR has been optimised. However, from the analysis in Section 3.4.5, we see that further tuning using the PAR has almost no impact on the performance when the PAR is dynamically adjusted. The recommended procedure for fine tuning of the parameter set is therefore to focus on the HMCR and move on to the HMS only when further tuning required. It is unlikely that further tuning would lead to significant performance gains.

Valuable insight into the consequences of disturbing the balanced relationship that memory consideration and pitch adjustment have was gained in the development of DCS. Even though DCS did not perform as well as hoped this insight led to the development of CHS and LEFHS that carefully considered this balanced relationship and performed much better for it. The key insight in this case was that memory consideration and pitch adjustment work together symbiotically. Pitch adjustment cannot be replaced by some other method that does not take into account the effect that memory consideration has on the search process. As we saw with DCS, reckless replacement of pitch adjustment can lead to over specialisation and even when one attempts to mitigate this problem through parameter tuning, the result tends to degrade to a random search with very poor performance.

The development of DCS also taught me how much harmony search depends on the fact that component values are *numerical* values (not simply labels) and that neighbouring solutions can be reached by incrementing and decrementing these values. By properly understanding and overcoming this relationship harmony search can even be used to solve labelling problems as CHS and LEFHS demonstrated.

The insights learned from DCS were extensively used in the development of CHS and LEFHS. Their development in turn led to other valuable insights particularly in the

relationship that the objective function has on the improvisation method. For example, due to the cadenza operation, TV regularisation tended to cause CHS to favour certain shapes above others causing an unfair bias in PSF solutions that often led to premature convergence to a local optimum. This was solved by changing the objective function and introducing TE regularisation. When the improvisation method was again changed in LEFHS the TE term had to be normalised in response to the way LEFHS gradually *grows* PSF solutions. Careful regularisation of the objective function was needed in both CHS and LEFHS and only by managing the effect that the objective function had on the novel improvisation schemes, could efficient optimisation be achieved.

Most of these insights are not specific to the application and are very valuable whenever one has to adapt harmony search to a specific problem. The value of the harmony filter, DCS, CHS and LEFHS is therefore certainly not only in their practical use but more so in the knowledge gained by their development. Even when not adapted to a specific problem harmony search has proven to be a very powerful and successful algorithm in multiple disciplines. However, as demonstrated in this thesis, when adapted to a specific problem, harmony search has the potential to perform better than the current state-of-the-art.

It is my hope that this thesis and the articles that it builds on will act as a basis for future research of harmony search in computer vision. In each of the four novel algorithms introduced in this thesis future research and possible methods of improving on the current version are mentioned and discussed and many improvements can be made to improve the speed and accuracy of these harmony search adaptations.

6.3 Future Work

As already stated, one of the primary goals of this research was to introduce harmony search to the computer vision community as a valuable addition to the set of algorithms commonly used in computer vision. The four algorithms presented in this thesis only represent an introduction to what harmony search can do for computer vision researchers and further research would certainly be beneficial especially in further development of these algorithms.

The harmony filter is an adaptation of harmony search and is certainly the least complex of the four adaptations introduced in this thesis. Many more sophisticated

versions of harmony search were introduced in Chapter 2 and it is reasonable to suspect that application of these improvements to visual tracking might result in even better performance than the basic harmony filter as presented. Another area, in the application of harmony search to visual tracking, where further research would be valuable is in efficient and accurate convergence detection. Inaccurate convergence detection is responsible for the extended iterations-to-convergence that were sometimes seen in the experiments of Section 3.4.8. The performance of the harmony filter would greatly benefit from better ways to detect convergence.

The DCS, CHS and LEFHS algorithms were much more ambitious in adapting harmony search to solve problems that it would normally not be able to solve. DCS was developed to find optimal correspondence sets but did not perform as well as hoped. The main reason for this poor performance is improper modelling of the problem in a way that is applicable to harmony search based optimisers. This insight was later applied during the development of CHS and LEFHS resulting in much better performance from algorithms that were equally (if not more) ambitious in their adaptation of harmony search. Future research in applying harmony search to the visual correspondence problem should focus on properly modelling the labelling problem so that it can be optimised using methods based on memory consideration and pitch adjustment.

The main limitation of the CHS and LEFHS algorithms is the size of the images that can practically be recovered. Due to the quadratic growth in the time in takes until convergence, only small images are currently considered. This problem is not unique to harmony search based algorithms and many state-of-the-art algorithms suffer from the same limitation. However, in some cases this limitation was overcome through clever partitioning of the problem. An example of this is the scheme that She n et al. uses and was mentioned in Section 5.6 as a possible way to improve the performance of LEFHS. This idea (or something similar to it) can also be applied to harmony search based algorithms. This could then result in an algorithm that boasts the 100% recovery accuary of CHS and LEFHS combined with the ability to recover much larger images than was possible before.

I hope that these suggestions on topics of future research would inspire other researchers to continue experimenting with harmony search and use it to improve on the results presented in this thesis. More importantly, I hope that these results will inspire other researchers to adapt harmony search for other uses in computer vision.

Bibliography

[1] K. S. Lee and Z. W. Geem, "A New Meta-Heuristic Algorithm for Continuous Engineering Optimization: Harmony Search Theory and Practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, pp. 3902–3933, 2005. 1, 23, 34

[2] Z. Geem, K. Lee, and Y. Park, "Application of Harmony Search to Vehicle Routing," *American Journal of Applied Sciences*, vol. 2(12), pp. 1552–1557, 2005. 1, 23, 50

[3] Z. W. Geem, "Particle-swarm harmony search for water network design," *Engineering Optimization*, vol. 41:4, pp. 297–311, April 2009. 1, 48, 113

[4] Z. W. Geem, J. H. Kim, and G. Loganathan, "A New Heuristic Optimization Algorithm: Harmony Seach," *Simulation*, vol. 76:2, pp. 60–68, 2001. 1, 11, 23, 34, 156, 171

[5] Z. W. Geem, ed., *Recent Advances In Harmony Search Algorithm*, vol. 270 of *Studies in Computational Intelligence*. Springer Berlin, 1st ed., 2010. 1, 11, 13, 23, 34, 38, 50, 171

[6] M. Minami, J. Agbanhan, and T. Asakura, "Manipulator visual servoing and tracking of fish using a genetic algorithm," *Industrial Robot: An International Journal*, vol. 26, no. 4, pp. 278–289, 1999. 1, 74

[7] Y. Morsly and M. S. Djouadi, "Genetic Algorithm combined to IMM approach for Tracking Highly Maneuvering Targets," *IAENG International Journal of Computer Science*, vol. 35, p. advanced online publication 19 February 2008, 2008.

[8] I. A. Sulistijono and N. Kubota, "Human Head Tracking Based on Particle Swarm Optimization and Genetic Algorithm," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 11, no. 6, pp. 681–687, 2007. 73, 74

[9] X. Zhang, S. Maybank, W. Hu, X. Li, and M. Zhu, "Sequential particle swarm optimisation for visual tracking," in *Proceedings of CVPR*, pp. 1–8, 2008. 74

[10] L. Dumas, M. El Rhabi, and G. Rochefort, "A robust method for deconvolution of barcode signals and non uniform illumination estimation," in *The International Conference on Image Processing (ICIP)*, 2010.

[11] Y.-W. Chen, T. Enokura, and Z. Nakao, "A hybrid ga/sa approach to blind deconvolution," in *Knowledge-Based Intelligent Electronic Systems, 2nd International Conference, KES 1998*, (Adelaide, South Australia), pp. 144–149, April 1998. 1, 148

[12] Z. W. Geem, "Improved Harmony Search from Ensemble of Music Players," *Lecture Notes in Computer Science*, vol. 4251, pp. 86–93, 2006. 1, 36, 37, 38, 113

[13] M. Mahdavi, M. Fesanghary, and E. Damangir, "An Improved Harmony Search Algorithm for Solving Optimization Problems," *Applied Mathematics and Computation*, vol. 188, pp. 1567–1579, 2007. 34, 38, 85

[14] C.-M. Wang and H. Yin-Fu, "Self-adaptive harmony search algorithm for optimization," *Expert Systems with Applications*, vol. 37, no. 4, pp. 2826–2837, 2009. 26, 40, 41, 42

[15] M. Fesanghary, M. Mahdavi, M. Minary-Jolandan, and Y. Alizadeh, "Hybridizing Harmony Search Algorithm With Sequential Quadratic Programming for Engineering Optimization Problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 197:33-40, pp. 3080–3091, 2008. 47, 113

[16] Z. Geem, "Novel derivative of harmony search algorithm for discrete design variables," *Applied Mathematics and Computation*, vol. 199:1, pp. 223–230, 2008. 19, 22

[17] M. Omran and M. Mahdavi, "Global-best Harmony Search," *Applied Mathematics and Computation*, vol. 198, pp. 643–656, 2008. 1, 33, 40, 47, 113

[18] C. Steger, M. Ulrich, and C. Wiedemann, *Machine Vision Algorithms and Applications*. Wiley-VCH, 1 ed., 2008. 3

[19] A. Davison, "Real-Time Simultaneous Localisation and Mapping with a Single Camera," in *IEEE International Conference on Computer Vision*, pp. 1403–1410, October 2003. 3

[20] R. Sim, P. Elinas, and J. J. Little, "A Study of the Rao-Blackwellised Particle Filter for Efficient and Accurate Vision-Based SLAM," *IJCV*, vol. 74, no. 3, pp. 303–318, 2007. 3, 66

[21] E. Geremia, B. H. Menze, O. Clatz, E. Konukoglu, A. Criminisi, and N. Ayache, "Spatial decision forests for ms lesion segmentation in multi-channel mr images.," *Medical image computing and computer-assisted intervention*, vol. 13, no. Pt 1, pp. 111–8, 2010. 3

[22] B. Menze, G. Langs, Z. Tu, and A. Criminisi, *Medical Computer Vision*, vol. 6533 of *Lecture Notes in Computer Science*. Springer Berlin, 1 ed., February 2011. 3

[23] J. Fourie, S. Mills, and R. Green, "Harmony filter: A robust visual tracking system using the improved harmony search algorithm," *Image Vision Computing*, vol. 28, pp. 1702–1716, December 2010. 5, 17, 31, 32, 50, 75, 183

[24] J. Fourie, S. Mills, and R. Green, "Visual tracking using the harmony search algorithm," in *Image and Vision Computing New Zealand. IVCNZ 23rd International Conference*, pp. 1–6, November 2008. 5, 31

[25] J. Fourie, R. Green, and S. Mills, "Visual tracking using harmony search," in *Recent Advances In Harmony Search Algorithm* (Z. W. Geem, ed.), vol. 270 of *Studies in Computational Intelligence*, chapter 4, pp. 37–50, Springer Berlin, 1st ed., 2010. 5, 75

[26] J. Fourie, S. Mills, and R. Green, "Directed correspondence search: Finding feature correspondences in images using the harmony search algorithm," in *Image and Vision Computing New Zealand (IVCNZ)*, November 2009. 5

[27] J. Fourie, S. Mills, and R. Green, "Counterpoint harmony search: An accurate algorithm for the deconvolution of binary images," in *Audio Language and Image Processing (ICALIP), 2010 International Conference on*, (Shanghai, China), pp. 1117 – 1122, November 2010. 6, 154, 177

[28] J. Fourie, S. Mills, and R. Green, "An accurate harmony search based algorithm for the blind deconvolution of binary images," in *Image and Vision Computing New Zealand (IVCNZ)*, (Queenstown, New Zealand), November 2010. 6, 169

[29] G. Polya, *How to Solve It.* Princeton, NJ: Princeton University Press, 1945. 10

[30] G. F. Luger, *Artificial Intelligence – Structures and Strategies for Complex Problem Solving.* Pearson Education Ltd., 2005. 10

[31] J. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI.: The University of Michigan Press, 1975. 11

[32] S. Kirkpatrick, J. Gelatt, C. D., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983. 11, 41, 49

[33] M. Dorigo, *Optimization, Learning and Natural Algorithms.* PhD thesis, Politecnico di Milano, Milano, Italie, 1992. 11

[34] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, vol. IV, pp. 1942–1948, 1995. 11, 40

[35] S. Das, A. Mukhopadhyay, A. Roy, A. Abraham, and B. K. Panigrahi, "Exploratory power of the harmony search algorithm: Analysis and improvements for global numerical optimization," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 41, no. 1, pp. 89 – 106, 2010. 22, 23, 24

[36] A. Mukhopadhyay, A. Roy, S. Das, and A. Abraham, "Population-variance and explorative power of harmony search: An analysis," in *Digital Information Management, 2008. ICDIM 2008. Third International Conference on*, pp. 775 –781, November 2008. 22

[37] H. Faure, "Good permutations for extreme discrepancy," *Journal of Number Theory*, vol. 42, pp. 47–56, 1992. 26

[38] J. Greblicki and J. Kotowski, "Analysis of the properties of the harmony search algorithm carried out on the one dimensional binary knapsack problem," in *Computer Aided Systems Theory - EUROCAST 2009: 12th International Conference, Las Palmas de Gran Canaria, Spain*, (Berlin, Heidelberg), pp. 697–704, Springer-Verlag, February 2009. 32, 33

[39] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer Verlag, 2005. 33

[40] M. F. Triola, *Elementary Statistics*. 75 Arlington Street, Suite 300, Boston, MA: Pearson Education Inc., 10'th ed., 2007. 37, 38

[41] N. Taherinejad, "Highly reliable harmony search algorithm," in *Circuit Theory and Design, 2009. ECCTD 2009. European Conference on*, pp. 818 –822, August 2009. 40

[42] D. Cvijovicacute and J. Klinowski, "Taboo Search: An Approach to the Multiple Minima Problem," *Science*, vol. 267, no. 5198, pp. 664–666, 1995. 41

[43] Q.-K. Pan, P. N. Suganthan, J. J. Liang, and M. F. Tasgetiren, "A local-best harmony search algorithm with dynamic subpopulations," *Engineering Optimization*, vol. 42, pp. 101–117, 2010. 43, 44

[44] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965. 45

[45] W. S. Jang, H. I. Kang, and B. H. Lee, "Hybrid simplex-harmony search method for optimization problems," in *IEEE Congress on Evolutionary Computation*, pp. 4157 –4164, June 2008. 45, 46

[46] L.-p. Li and L. Wang, "Hybrid algorithms based on harmony search and differential evolution for global optimization," in *GEC '09: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, (New York, NY, USA), pp. 271–278, ACM, 2009. 46

[47] R. Storn and K. Price, "Differential evolution a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, December 1997. 46

[48] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numerica*, vol. 4, no. -1, pp. 1–51, 1995. 48

[49] S. Chib and E. Greenberg, "Understanding the metropolishastings algorithm," *American Statistician*, vol. 49, no. 4, pp. 327–335, 1995. 49

[50] H. Jiang, Y. Liu, and L. Zheng, "Design and simulation of simulated annealing algorithm with harmony search," in *Advances in Swarm Intelligence* (Y. Tan, Y. Shi, and K. Tan, eds.), vol. 6146 of *Lecture Notes in Computer Science*, pp. 454–460, Springer Berlin Heidelberg, 2010. 49

[51] O. Alia, R. Mandava, D. Ramachandram, and M. Aziz, "Dynamic fuzzy clustering using harmony search with application to image segmentation," in *Signal Processing and Information Technology (ISSPIT), 2009 IEEE International Symposium on*, pp. 538 –543, December 2009. 50, 144, 184

[52] Z. Geem, "Harmony search algorithm for solving sudoku," in *Knowledge-Based Intelligent Information and Engineering Systems* (B. Apolloni, R. Howlett, and L. Jain, eds.), vol. 4692 of *Lecture Notes in Computer Science*, pp. 371–378, Springer Berlin / Heidelberg, 2010.

[53] Z. W. Geem and J.-Y. Choi, "Music composition using harmony search algorithm," in *Proceedings of the 2007 EvoWorkshops*, (Berlin, Heidelberg), pp. 593–600, Springer-Verlag, 2007. 50

[54] K. Nummiaro, E. Koller-Meier, and L. V. Gool, "An adaptive color-based particle filter," *Image and Vision Computing*, vol. 21, no. 1, pp. 99 – 110, 2003. 53, 63, 72, 77

[55] M. Greiffenhagen, V. Ramesh, D. Comaniciu, and H. Niemann, "Statistical modeling and performance characterization of a real-time dual camera surveillance system," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2, pp. 335 –342, August 2000. 53

[56] U. Handmann, T. Kalinke, C. Tzomakas, M. Werner, and W. von Seelen, "An Image Processing System for Driver Assistance," *Image Vision Comput.*, vol. 18, no. 5, pp. 367–376, 2000. 53

[57] M. J. Black and A. D. Jepson, "A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions," in *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume I*, (London, UK), pp. 909–924, Springer-Verlag, 1998. 53

[58] G. Klein and T. Drummond, "Robust visual tracking for non-instrumented augmented reality," in *ISMAR '03: Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, (Washington, DC, USA), p. 113, IEEE Computer Society, 2003. 53

[59] B. Menser and M. Brunig, "Face detection and tracking for video coding applications," in *Signals, Systems and Computers, 2000. Conference Record of the Thirty-Fourth Asilomar Conference on*, vol. 1, pp. 49 –53, October 2000. 53

[60] C. Rasmussen and G. D. Hager, "Probabilistic data association methods for tracking complex visual objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 560–576, 2001. 54

[61] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics.* The MIT Press Cambridge, Massachusetts, 2005. 54, 63, 68

[62] B. Ristic, S. Arulampalm, and N. Gordon, *Beyond the Kalman filter : particle filters for tracking applications.* London, England: Artech House, Boston,, 2004. 55, 66, 68

[63] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960. 55, 56

[64] M. I. Jordan, *Learning in graphical models.* Cambridge, MA, USA: MIT Press, 1999. 56

[65] B. Anderson and J. Moore, *Optimal Filtering.* New Jersey, U.S.A: Prentence-Hall, 1979. 58

[66] E. Brookner, *Tracking and Kalman Filtering Made Easy.* Wiley Interscience, 1998. 59

[67] S. Julier and J. K. Uhlmann, "A general method for approximating nonlinear transformations of probability distributions," tech. rep., Department of Engineering Science, University of Oxford, 1996. 60, 61, 62, 63

[68] A. Doucet, R. van der Merwe, N. de Freitas, and E. Wan, "The Uncented Particle Filter," tech. rep., Cambridge University Engineering Department, 2000. 60, 64, 65, 66, 68, 70

[69] P. Li, T. Zhang, and B. Ma, "Unscented Kalman filter for visual curve tracking," *Image and Vision Computing*, vol. 22, pp. 157–164, 2004. 63

[70] A. Doucet, N. de Freitas, and N. Gorden, *Sequential Monte Carlo Methods in Practice.* Springer-Verlag New York, Inc., 2001. 63, 64, 66, 67, 68

[71] M. Isard and A. Blake, "Condensation – Conditional Density Propagation for Visual Tracking," *IJCV*, vol. 29, no. 1, pp. 5–28, 1998. 63

[72] S. Thrun, "Particle Filters in Robotics," in *Proceedings of Uncertainty in Artificial Intelligence*, 2002. 63

[73] H. Andreasson, A. Treptow, and T. Duckett, "Localization for Mobile Robots using Panoramic Vision, Local Features and Particle Filter," in *IEEE International Conference on Robotics and Automation (ICRA 2005)*, (Barcelona, Spain), 2005.

[74] L. Marchetti, G. Grisetti, and L. Iocchi, "A Comparative Analysis of Particle Filter Based Localization methods," in *RoboCup*, pp. 442–449, 2006. 63

[75] H. Durrant-Whyte and T. Bailey, "Simultaneous Localization and Mapping: Part I." This tutorial can be found at `http://automation.tkk.fi/AS-84-3145/AS-84-3145-en?action=download&upname=SLAMTutorial.pdf`, 2006. 63

[76] A. Doucet, N. Gordon, and V. Krishnamurthy, "Particle filters for state estimation of jump markov linear systems," tech. rep., Cambridge university engineering department, 1999. 66

[77] G. Grisetti, G. D. Tipaldi, C. Stachniss, W. Burgard, and D. Nardi, "Fast and Accurate SLAM with Rao-Blackwellized Particle Filters," *Robot. Auton. Syst.*, vol. 55, no. 1, pp. 30–38, 2007. 66

[78] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges," in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pp. 1151–1156, 2003.

[79] C. Stachniss, G. Grisetti, W. Burgard, and N. Roy, "Analyzing Gaussian Proposal Distributions for Mapping with Rao-Blackwellized Particle Filters," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (San Diego), October 2007. 66

[80] N. Gordon, *Bayesian Methods for Tracking.* PhD thesis, University of London, 1993. 68

[81] T. Kailath, "The Divergence and Bhattacharyya Distance Measures in Signal Selection," *IEEE Transactions on Communication Technology*, vol. 15, no. 1, pp. 52–60, 1967. 72

[82] T. Chen, R. Luo, and T. Hsiao, "Visual tracking using adaptive color histogram model," in *The 25th Annual Conference of the IEEE Industrial Electronics Society*, vol. 3, pp. 1336 – 1341, 1999. 73

[83] M. Swain and D. Ballard, "Indexing Via Color Histograms," in *Proceedings of The Third International Conference on Computer Vision*, pp. 390–396, 1990. 73

[84] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-Based Object Tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–575, 2003. 73

[85] H. Guyen, G. Kogut, R. Barua, A. Burmeister, N. Pezeshkian, D. Powell, N. Farrington, M. Wimmer, B. Cicchetto, C. Heng, and V. Ramirez, "A Segway RMP-Based Robotic Transport System," in *SPIE Proc. 5609: Mobile Robots XVII*, 2004. 77

[86] J. Sattar, "A Visual Servoing System for an Amphibious Legged Robot," Master's thesis, McGill University, Montréal, Québec, 2005. 77

[87] A. Zeileis, K. Hornik, and P. Murrell, "Escaping RGBland: Selecting colors for statistical graphics," *Computational Statistics and Data Analysis*, vol. 53, pp. 3259–3270, July 2009. 77

[88] J. Placzek, *Orthopaedic Physical Therapy Secrets*. Philadelphia, U.S.A: Hanley & Belfus, 2006. 87

[89] "EC funded CAVIAR project/ist 2001 37540." `http://homepages.inf.ed.ac.uk/rbf/CAVIAR/`, July 2008. 97

[90] L. Hong-qi, L. Li, T. Kim, and X. Shao-long, "An Improved PSO-based of Harmony Search for Complicated Optimization Problems," *International Journal of Hybrid Information Technology*, vol. Vol 1., No. 1, pp. 57–64, 2008. 113

[91] D. Nister, "Preemptive ransac for live structure and motion estimation," in *Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set*, 2003. 116

[92] J. Kim, V. Kolmogorov, and R. Zabih, "Visual correspondence using energy minimization and mutual information," in *International Conference on Computer Vision (ICCV)*, 2003. 116, 129

[93] C. Silpa-Anan and R. Hartley, "Visual localization and loop-back detection with a high resolution omnidirectional camera," in *The 6'th Workshop on Omnidirectional vision, Camera Networks and Non-classical cameras*, 2005. 116, 117

[94] D. Scharstein, R. Szeliski, and R. Zabih, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," in *Stereo and Multi-Baseline Vision, 2001. (SMBV 2001). Proceedings. IEEE Workshop on*, pp. 131 –140, 2001. 116

[95] D. Scaramuzza, F. Fraundorfer, and R. Siegwart, "Real-time monocular visual odometry for on-road vehicles with 1-point ransac," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pp. 4293 –4299, may. 2009. 117

[96] E. Angel, *Interactive Computer Graphics: A Top-Down Approach using OpenGL (4th Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005. 119

[97] J. Shi and C. Tomasi, "Good features to track," in *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pp. 593 – 600, 1994. 120

[98] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Nov 2004. 120

[99] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia. 120, 137

[100] T. Botterill, *Visual Navigation for Mobile Robots using the Bag-of-Words Algorithm*. PhD thesis, University of Canterbury, Christchurch, New Zealand, 2010. 120, 121, 125

[101] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Fourth Alvey Vision Conference*, pp. 147–151, 1988. 121

[102] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *In European Conference on Computer Vision*, pp. 430–443, 2006. 121

[103] Brown, Matthew, Szeliski, Richard, and Winder, Simon, "Multi-Image Matching Using Multi-Scale Oriented Patches," *Computer Vision and Pattern Recognition, IEEE Computer Society Conferenceon*, vol. 1, pp. 510–517, 2005. 123

[104] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, p. 2006, 2006. 123

[105] J. S. Beis and D. G. Lowe, "Indexing without invariants in 3d object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 1000–1015, 1999. 123

[106] P. H. S. Torr and D. W. Murray, "The development and comparison of robust methodsfor estimating the fundamental matrix," *Interantional Journal of Computer Vision*, vol. 24, no. 3, pp. 271–300, 1997. 123

[107] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[108] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, UK: Cambridge University Press, second ed., 2003. 123, 135

[109] P. H. S. Torr and A. Zisserman, "MLESAC: a new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 138–156, 2000. 124

[110] P. J. Rousseeuw and A. M. Leroy, *Robust regression and outlier detection*. New York, NY, USA: John Wiley & Sons, Inc., 1987. 124

[111] O. Chum and J. Matas, "Matching with PROSAC - progressive sample consensus," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)* (C. Schmid, S. Soatto, and C. Tomasi, eds.), vol. 1, (Los Alamitos, USA), pp. 220–226, IEEE Computer Society, June 2005. 125

[112] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 1222–1239, November 2001. 125, 126, 128, 129

[113] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentence-Hall, 1993. 125

[114] Y. Gdalyahu, D. Weinshall, and M. Werman, "Self-organization in vision: Stochastic clustering for image segmentation, perceptual grouping, and image database organization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 1053–1074, 2001. 125

[115] O. Veksler, "Image segmentation by nested cuts," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 339–344, 2000. 125

[116] H. Ishikawa and D. Geiger, "Occlusions, discontinuities, and epipolar lines in stereo," in *5th European Conference on Computer Vision (ECCV)*, pp. 89–96, 1998. 129

[117] S. Roy and I. J. Cox, "A maximum-flow formulation of the n-camera stereo correspondence problem," in *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98, (Washington, DC, USA), pp. 492–, IEEE Computer Society, 1998. 129

[118] D. S. C. Biggs, *Accelerated Iterative Blind Deconvolution.* PhD thesis, The Department of Electrical and Electronic Engineering, University of Auckland, New Zealand, 1998. 148, 155, 156

[119] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition).* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. 148, 155

[120] D. Kundur and D. Hatzinakos, "Blind image deconvolution," *IEEE Signal Processing Magazine*, vol. 13 no.5, pp. 43–64, May 1996. 148, 149

[121] T. Chan and C. Wong, "Total variation blind deconvolution," *IEEE Transactions on Image Processing*, vol. 7, pp. 370–375, March 1998. 155

[122] J. Zhang, Q. Zhang, and G. He, "Blind deconvolution: multiplicative iterative algorithm," *Optics Letters*, vol. 33, no. 1, pp. 25–27, 2008. 148

[123] T.-H. Li and K.-S. Lii, "Deblurring two-tone images by a joint estimation approach using higher-order statistics," in *SPWHOS '97: Proceedings of the 1997 IEEE Signal Processing Workshop on Higher-Order Statistics (SPW-HOS '97)*, (Washington, DC, USA), p. 108, IEEE Computer Society, 1997. 148

[124] E. Y. Lam and J. W. Goodman, "Iterative statistical approach to blind image deconvolution," *J. Opt. Soc. Am. A*, vol. 17, no. 7, pp. 1177–1184, 2000. 148, 149

[125] J. Keuchel, C. Schnrr, C. Schellewald, and D. Cremers, "Binary partitioning, perceptual grouping, and restoration with semidefinite programming," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 1364–1379, 2003. 149, 154

[126] T. Chan and C. Wong, "Total variation blind deconvolution," *IEEE Transactions on Image Processing*, vol. 7, pp. 370–375, March 1998. 149, 155

[127] M. R. Banham and A. I. Katsaggelos, "Digital image restoration," *IEEE Signal Processing Magazine*, vol. 14 no.2, pp. 24–41, March 1997. 149

[128] E. Lam, "Blind bi-level image restoration with iterated quadratic programming," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 54, no. 1, pp. 52–56, 2007. 149, 153

[129] Y. Shen, E. Y. Lam, and N. Wong, "Restoration of binary images using positive semidefinite programming," *Optics Letters*, vol. 32, no. 2, pp. 121–123, 2007. 149, 167, 182

[130] A. Raj and R. Zabih, "A graph cut algorithm for generalized image deconvolution," in *Proceedings of the International Conference on Computer Vision 2005 (ICCV'05)*, pp. 1048–1054, 2005. 150

[131] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C*. Cambridge Press, 1992. 150

[132] Y. Shen, E. Y. Lam, and N. Wong, "Robust binary image deconvolution with positive semidefinite programming," *IAENG International Journal of Applied Mathematics*, vol. 36, no. 1, pp. 1–8, 2007. 151, 152, 167, 177, 179, 180, 182

[133] T. F. Chan, S. Esedoglu, and M. Nikolova, "Finding the global minimum for binary image restoration," in *The International Conference on Image Processing (ICIP)*, pp. 121–124, 2005. 154

[134] M. Lun and L. Guisheng, "A novel autofocus algorithm based on maximum total variation criteria for sar images," *Frontiers of Electrical and Electronic Engineering in China*, vol. 2:3, pp. 298–301, 2007. 155

[135] H. Wang, L. Tang, and Z. Cao, "An image reconstruction algorithm based on total variation with adaptive mesh refinement for ect," *Flow Measurement and Instrumentation*, vol. 18, no. 5-6, pp. 262 – 267, 2007. Process Tomography and Flow Visualization. 155

[136] T. Niwa and M. Tanaka, "Analysis on the island model parallel genetic algorithms for the genetic drifts," in *SEAL'98: Selected papers from the Second Asia-Pacific*

Conference on Simulated Evolution and Learning on Simulated Evolution and Learning, (London, UK), pp. 349–356, Springer-Verlag, 1999. 158

[137] B. Artyushenko, "Analysis of global exploration of island model genetic algorithm," in *CAD Systems in Microelectronics, 2009. CADSM 2009. 10th International Conference*, pp. 280 –281, February 2009. 158

Harmony Filter Pseudo Code

input : G_{idleMax} = the maximum idle iterations allowed
input : $T_{\text{precision}}$ = the precision threshold
input : T_{quality} = the the quality threshold
output: True if convergence has been detected else false

// $I_{\texttt{best}}, I_{\texttt{worst}}$ are the best and worst candidates in the HM
$d \leftarrow \sqrt{(I_{\text{best}}[0] - I_{\text{worst}}[0])^2 + (I_{\text{best}}[1] - I_{\text{worst}}[1])^2}$
if $d < T_{\text{precision}}$ and $I_{\text{best}} > T_{\text{quality}}$ **then** // check spatial convergence
  RETURN TRUE
end
if idleIters $\geq G_{\text{idleMax}}$ **then** // check idle iterations
  RETURN TRUE
end
RETURN FALSE

Algorithm .1: The Harmony Filter convergence test

input : H_{target} = the template histogram
input : $G_{\max}$ = the maximum iterations allowed
input : σ_a = the acceleration variance
input : σ_s = the scale variance
input : $\mathbf{x}_{t-1}^{\text{best}}$ = the optimal solution vector from the previous frame
input : T_{lost} = the lost tracker weight threshold
output: $\mathbf{x}_t^{\text{best}}$ = the current optimal solution

// Initialisation of the HM
foreach $i \in [1, 2, \ldots, \text{HMS}]$ **do**
    $C_s \leftarrow \mathbf{x}_{t-1}^{\text{best}}[s] \times r$ // where $r \backsim U(1, \sigma_s)$
    $a_x \leftarrow \mathcal{N}(0, \sigma_a)$ // a is independently sampled from $\mathcal{N}(0, \sigma_a)$
    $a_y \leftarrow \mathcal{N}(0, \sigma_a)$ // a is independently sampled from $\mathcal{N}(0, \sigma_a)$
    $C_x \leftarrow \mathbf{x}_{t-1}^{\text{best}}[x] + \mathbf{x}_{t-1}^{\text{best}}[\dot{x}] + \frac{a_x}{2}$
    $C_y \leftarrow \mathbf{x}_{t-1}^{\text{best}}[y] + \mathbf{x}_{t-1}^{\text{best}}[\dot{y}] + \frac{a_y}{2}$
    $C_{\dot{x}} \leftarrow \mathbf{x}_{t-1}^{\text{best}}[\dot{x}] + a_x$
    $C_{\dot{y}} \leftarrow \mathbf{x}_{t-1}^{\text{best}}[\dot{y}] + a_y$
    $H_C \leftarrow \text{calcHist}(C_x, C_y, s)$ // get histogram of $[C_x, C_y, C_s]$ area.
    $w \leftarrow \text{Bhatt}(H_C, H_{\text{target}})$ // weigh H_C using Bhattacharrya
    $\mathbf{C} \leftarrow [C_x, C_y, C_{\dot{x}}, C_{\dot{y}}, C_s, w]$
    $\text{HM}[i] \leftarrow \mathbf{C}$
end
$w_{\text{best}} \leftarrow \max(\text{HM})[w]$ // get weight of best candidate
$w_{\text{worst}} \leftarrow \min(\text{HM})[w]$ // get weight of worst candidate
// The main HS loop
foreach $g \in [1, 2, \ldots, G_{\max}]$ **do**
    // Update PAR and FW according to eqations (2.34)-(2.36)
    if $w_{\text{best}} < T_{\text{lost}}$ **then** // lost tracker detection
        HMCR $\leftarrow 0.0$
    end
    $I \leftarrow \text{getImprov}(HM)$ // improvise using algorithm .3
    if $I[6] > w_{\text{worst}}$ **then** // update the HM
        $\min(\text{HM}) \leftarrow I$
        $w_{\text{best}} \leftarrow \max(\text{HM}[w])$
        $w_{\text{worst}} \leftarrow \min(\text{HM}[w])$
    else
        idleIters = idleIters + 1 // this is an idle iteration
    end
    // Check for convergence
    if checkConvergence(idleIters, HM) = True **then** // see algorithm .1
        BREAK FROM LOOP
    end
end
$\mathbf{x}_t^{\text{best}} = \max(\text{HM})$

Algorithm .2: The Harmony Filter

input : The initialised HM
input : $\mathbf{x}_{t-1}^{\text{best}}$ = the optimal solution vector from the previous frame
input : σ_a = the acceleration variance
input : σ_s = the scale variance
output: The new improvisation I

$I \leftarrow []$ // An empty vector of 6 components
foreach $j \in [3, 4, 5]$ **do** // consider only the velocities and scale
  if $U(0,1) > HMCR$ **then** // random selection
    if $HMCR = 0$ **then** // is the tracker lost
      if $j = 5$ **then** // consider the scale
        $I[j] \leftarrow 1$ // reset scale to unity
      else
        $I[j] \leftarrow 0$ // zero the velocity
        $I[j-2] \leftarrow r$
        // r is a random pixel
        // uniformly sampled from the whole frame
      end
    else
      if $j = 5$ **then** // consider the scale
        $I[j] \leftarrow \mathbf{x}_{t-1}^{\text{best}}[j] \times r$ // where $r \backsim U(-\sigma_s, \sigma_s)$
      else
        $I[j] \leftarrow \mathbf{x}_{t-1}^{\text{best}}[j] + r$ // where $r \backsim U(-\sigma_a, \sigma_a)$
      end
    end
  else // memory consideration
    index $\leftarrow U(1, \text{HMS})$ // get a random member of the HM
    $I \leftarrow$ HM[index]
    if $U(0,1) < PAR$ **then** // pitch adjustment
      $I[j] \leftarrow I[j] + r$ // where $r \backsim U(-FW, FW)$
  end
end
if $HMCR \neq 0.0$ **then** // update position from velocity
  $I[1] \leftarrow \mathbf{x}_{t-1}^{\text{best}}[1] + I[3]$
  $I[2] \leftarrow \mathbf{x}_{t-1}^{\text{best}}[1] + I[4]$
end
$H_I \leftarrow \text{calcHist}(I[1], I[2], I[5])$ // get histogram of $I[1], I[2], I[5]$ area.
$w \leftarrow \text{Bhatt}(H_I, H_{\text{target}})$ // weigh H_I using Bhattacharrya
$I[6] \leftarrow w$
RETURN I

Algorithm .3: The Harmony Filter improvisation step

DCS Pseudo Code

```
   input  : The HM
   input  : M = the number of components in the solution vector
   output: The new improvisation I
 1 foreach i ∈ [1, M] do
 2     if U(0, 1) > HMCR then                          // random selection
 3         I[i] ← V(r)                                      // r ⇀ U(0, |V|)
 4         if I[i] ≠ φ then
 5             V ← V − I[i]          // − is the remove from set operator
 6         end
 7     else                                          // memory consideration
 8         while true do                 // loop until options are exhausted
 9             I[i] ← HM[j][i]                                  // j ⇀ U(1, HMS)
10             if I[i] ∈ V then
11                 if I[i] ≠ φ then
12                     V ← V − I[i]
13                 end
14                 BREAK FROM LOOP
15             end
16         end
17     end
18 end
```

Algorithm .4: The DCS improvisation step

input : $G_{\max}$ = the maximum iterations allowed
input : SR = the swap rate
output: C_{RA}^{best} = the optimal correspondence set

$w_{\text{best}} \leftarrow \max(\text{HM})[w]$ // get weight of best candidate
$w_{\text{worst}} \leftarrow \min(\text{HM})[w]$ // get weight of worst candidate
HM $\leftarrow$ DCSInit(HM) // see Algorithm .6
// The main DCS loop
foreach $g \in [1, 2, \ldots, G_{\max}]$ **do**
  $\mathbf{V} \leftarrow F_A$ // initialise $\mathbf{V}$ to include all features
  if idleIters $\geq SR$ **then** // trigger swap operation
    foreach $d \in [1, 2, \ldots, M]$ **do**
      $I[d] \leftarrow \max(\text{HM})[d]$ // use the best candidate in the HM
      $\mathbf{V} \leftarrow \mathbf{V} - I[i]$ // this feature cannot be used again
    end
    // Get the fitness weight, worst and second worst indices using Algorithm .7
    $[w, \text{worstIndex}, \text{worstIndex2}] \leftarrow \text{getFitness}(I)$
    $I[M] \leftarrow w$
    // Swap values at worst and second worst index
    $\text{Swap}(I[\text{worstIndex}], I[\text{worstIndex2}])$
    $w \leftarrow \text{getFitness}(I)$ // calculate post-swap weight
    if $w_{\text{worst}} \geq w$ **then** // swap made no improvement
      // swap back and replace with random swap
      $\text{Swap}(I[\text{worstIndex}], I[\text{worstIndex2}])$
      $\text{worstIndex2} \leftarrow \mathbf{V}(j)$ // $j \backsim U(0, |\mathbf{V}|)$
      $\text{Swap}(I[\text{worstIndex}], I[\text{worstIndex2}])$
    end
    $I[M] \leftarrow w$
  else // DCS standard improvisation
    $I \leftarrow \text{DCSImprov(HM)}$ // see Algorithm .4
    $w \leftarrow \text{getFitness}(I)$
  end
  if $w > w_{\text{worst}}$ **then** // update the HM
    $\min(\text{HM}) \leftarrow I$
    $w_{\text{best}} \leftarrow \max(\text{HM}[w])$
    $w_{\text{worst}} \leftarrow \min(\text{HM}[w])$
  else
    $\text{idleIters} = \text{idleIters} + 1$ // this is an idle iteration
  end
end
$C_{RA}^{\text{best}} = \max(\text{HM})$

Algorithm .5: The Directed Correspondence Search algorithm

input : F_R = features from the reference image where $|F_R| = M$
input : F_A = features from the alternate image where $|F_A| = N$
input : matchThreshold = the minimum SSD score for a valid match
output: the initialised HM

```
// generate prior
foreach i ∈ [1, 2, ..., M] do
    bestScore ← ∞                          // initialise best score to max
    bestIndex ← 0                          // initialise best index
    foreach j ∈ [1, 2, ..., N] do
        s ← SSD(F_R[i], F_A[j])
        if s < bestScore then              // currently the best match
            bestScore ← s
            bestIndex ← j
        end
    end
    if bestScore < matchThreshold AND bestScore ∈ V then
        // this is a valid match
        prior[i] ← bestIndex
        V ← V + bestIndex                  // + here means set addition
    else                                   // no observation was found
        prior[i] ← φ
    end
end
HM[0] ← prior                              // initialise HM with prior
HM[1] ← prior
foreach i ∈ [2, 3, ..., HMS] do
    V ← F_A                    // initialise V to include all features
    foreach j ∈ [0, 1, ..., M] do
        if j = M then                          // this is the weight
            HM[i][j] = getFitness(HM[i])
        else                               // get random value from V
            HM[i][j] = r                                  // r ∽ U(0, |V|)
            V ← V − r              // this feature cannot be used again
        end
    end
end
```

Algorithm .6: The DCS initialisation step

input : I = the candidate correspondence set
input : matchThreshold = the minimum SSD score for a valid match
input : ρ = a small constant factor penalising unmatched features
input : α = the importance multiplier for the variance term
input : β = the importance multiplier for the matches term
input : τ = the importance multiplier for the matches term
output: s = the fitness score
output: worstIndex the index of the worst component
output: worstIndex2 the index of the second worst component

foreach $i \in [1, 2, \ldots, M]$ **do**
  if $I[i] = \phi$ **then** `// this feature is unmatched`
    compScore $\leftarrow$ matchThreshold $+ \rho$
  else `// get relative motion vector and SSD measure`
    motionVectors$[i] \leftarrow F_R[i] - F_A[I[i]]$
    totalMatches $\leftarrow$ totalMatches $+ 1$
    motionSum $\leftarrow$ motionSum $+$ motionVectors$[i]$
    squareSum $\leftarrow$ squareSum $+$ motionVectors$[i] \times$ motionVectors$[i]$
    compScore $\leftarrow$ SSD$(F_R[i], F_A[I[i]])$
  end
  SSDScore $\leftarrow$ SSDScore $+$ compScore
end
motionMean $\leftarrow \frac{\text{motionSum}}{\text{totalMatches}}$
motionVariance $\leftarrow \frac{\text{squareSum} - \text{motionSum}}{\text{totalMatches} - 1}$
motionScore $\leftarrow ||$motionVariance$||$
worstIndex $\leftarrow 0$ worstIndex2 $\leftarrow 0$
worstScore $\leftarrow 0$ worstScore2 $\leftarrow 0$
foreach $i \in [1, 2, \ldots, M]$ **do** `// find the worst components`
  if $I[i] \neq \phi$ **then** `// only consider valid matches`
    errorDistance $\leftarrow ||$motionVectors$[i] -$ motionMean$||$
  end
  if errorDistance $>$ worstScore **then** `// update the current worst component`
    worstScore2 $\leftarrow$ worstScore
    worstIndex2 $\leftarrow$ worstIndex
    worstScore $\leftarrow$ errorDistance
    worstIndex $\leftarrow i$
  else if errorDistance $>$ worstScore2 **then**
    worstScore2 $\leftarrow$ errorDistance
    worstIndex2 $\leftarrow i$
  end
end
$s = -(\text{SSDScore} + \alpha$ motionScore $+ \beta e^{-\text{totalMatches}\,\tau})$

Algorithm .7: The DCS objective function